$2.00

kilobaud T.M.

The Small Computer Magazine

ISSUE # 3

March 1977

Articles

Features

I Print "Publisher's Remarks" 10 No End Run

Wayne Green

THOSE MISERABLE ADS

While most of us enjoy reading the ads in our hobby magazines just as much as we do the editorial matter (articles), there are a few grouches who grumble about the ads and begrudge them page space. Perhaps I can put this into perspective for you.

Without the ads in *Kilobaud* the price of a subscription would have to double and the number of pages would have to be cut back. That's the economic importance of the ads as far as publishing is concerned. Each ad page makes it possible to print another two pages of articles, so advertisers are footing a good deal of the bill for the magazine for you.

Let's suppose for the moment that none of the hobby computer magazines accepted advertising. How would manufacturers sell their products? They would have to go to direct mail selling and that is historically far more costly per sale than selling through magazine ads. You, the customer, would have to pay for this inefficiency, so you'd pay more two ways . . . magazine cost and product cost.

While it isn't cheap to advertise in *Kilobaud* (ads run about $1000 per page), it still is the most economical way of getting sales. An ad which is written well will normally bring in ten to twenty times its cost. New firms find that it takes about six months of advertising before customers develop enough confidence to buy freely – which is why consistent advertising pays good dividends.

I first learned about consistent advertising when I started a mail order book business. I ran a half page ad in a magazine and got four sales from it. *Four!* Since I was on contract I let the ad run with a sinking sensation. The second ad brought in about 50 orders – not so bad, but no brass ring. By the time my six-insertion contract had run out I was handling over 400 orders a month from the same ad which had brought four orders the first time out. Since that time I've watched this phenomenon repeat itself over and over for other firms. It takes a lot of guts to hang in there and wait for the customers to get confidence in you . . . and not a little investment.

What makes an ad well written? There are a lot of books which explain what has been definitely discovered about advertising. It is no science, by any means, but some rules have evolved which are foolhardy to ignore. I'll be writing about the elements of advertising in the *Kilobaud* newsletters which are sent to the industry and club secretaries.

One fundamental of an ad is to make sure it is noticed. Some anti-male chauvinists get up tight over Tri-Tek's Ampl'Anny, but I doubt if anyone doesn't notice her. Score one for Tri-Tek.

The next step is to make sure that your ad is read . . . and this usually means a good catchy bold phrase . . . like Paia's "Can Your Computer Make Music?" Busy surplus-type ads get readership too, promising savings if you read the tiny print. I don't know if anyone has been able to make the type so small that people wouldn't read it . . . several have tried.

Whether the product is a $10,000 computer system or a $1 line cord, the ad should do the selling. If a customer has to send for further information before he is sure he wants to buy, the ad is a loser. Take a look at the ads and you'll see why some firms are doing fine and others hurting.

An ad is like a store window for the firm . . . a messy ad will scare off the customer . . . an over elaborate ad will indicate higher prices. Reading the Godbout ads you can almost see the warehouse with no nonsense cardboard boxes full of ICs and parts, the reason Bill is able to sell choice stuff for such a low price.

One manufacturer wrote to say he thought it was none of my darned business what problems readers had with manufacturers. I disagree with him. I ask that readers who have found manufacturers unresponsive write out the complete story in detail, giving dates, and send it to the firm . . . with a copy to me. In most cases this procedure tends to iron out the problems.

I've run into some lulus in the past . . and not so distant past. One firm had a well advertised product on the market which just flat wouldn't work. A few of the more engineering minded customers were able to redo the unit so it would work, but the factory was rarely able to duplicate this feat. They threatened to sue me when I refused to carry their ads.

If you find you are unable to understand an ad, why not drop me a line and perhaps we can help the firm to put its story across better and perk up its sales. There are some very fine products out there which aren't selling as well as they should.

HOW MANY HOBBY COMPUTERS ARE THERE . . . REALLY?

You probably enjoy reading articles in newspapers and news magazines about how fantastically the computer hobby is growing. I love reading inspirational stuff like that, even though I know it's mostly pleasantly scented hot air. The hype is ripe.

One of the pitchmen recently took to the road telling all who would listen that the field was "totally explosive." It's growing, unquestionably, but I'd hardly characterize it as totally explosive.

We see new clubs being listed in the magazines every month and new dealers. We see more and more microcomputer systems being advertised. We hear second hand that such and such a firm is selling over a million dollars a month (not Mits, by the way) . . . and we are enthusiastic. Now and then we get a glimpse of the other side of the coin . . . this firm is looking to sell out . . . that one was bought . . . and some of the dealers are having a tough row to hoe. What, we may wonder, is reality?

Some dealers confided to me that Mits had sold over 15,000 Altairs during 1975 and expected to at least double that during 1976. That's pretty good for a field which didn't even exist before 1975.

Well, we can expect a lot from Mits since they were first, but what about Imsai? They certainly are highly visible these days and must be pushing Mits a bit. A parts jobber, whom I respect, confided that he could tell quite closely from the parts the two firms were using how many systems they were selling . . . and that Imsai was outselling Mits seven to one. Incredible! That would put about 200,000 Imsai systems out there. Even though manufacturers seem to agree that only about one half of the systems are being bought by hobbyists, with the others going to OEMs and other commercial users, this is still a great start for a new industry.

That would give us about 250,000 systems manufactured by Mits and Imsai so far . . . plus God knows how many from the rest of the industry: Southwest Tech, Wave Mate, Digital Group, Sphere, Intelligent Systems, Jolt, Apple, M&R, Cromemco, ProcTech, Quay, TDL, etc. If we figure about $2000 per system that could put the dollar volume of the industry in the $1 billion bracket.

Whoa! Let's just wait a minute here. I made a round of the industry in August 1975 when I was starting *Byte Magazine* . . . and again in August 1976 for the I/O section of 73 (as a matter of fact, it was during the 1976 trip that I was convinced by the manufacturers that I should start *Kilobaud*) . . . but I sure didn't see any $1 billion industry out there. I saw a few small firms and a bunch of very small firms. Obviously we could use some hard data on what is going on . . . really.

Getting the facts directly from the manufacturers is out of the question . . . most of 'em are feeling guilty for not having as much in sales as they think everyone else has and don't want anyone to know how slow things really are. So how can we get a handle on what's been going on?

We've had two inputs recently which may throw some light on the actual size of the hobby computer market. I've done some homework on this and a lot of calculations . . . it may interest you. One input is the thousands of *Kilobaud* Sweepstakes cards which have been coming in. There is a little poll on the card which asks what system the respondent has running and what he may be working on. After a couple of days collating the dope from these cards I have a general idea of the percentages of various computer systems in hobby hands.

The poll answers are not exact . . . undoubtedly some chaps with Altairs just admitted to 8080 systems . . . some Southwest Techers just indicated 6800 systems . . . so there is a bit of latitude.

Percentages tell us part of the story, but what we really need to pin things down is the sales figures for just one of the firms. Luckily we have just that . . . I think. When the papers announced the sale of Mits to Pertec they gave the sales figures for 1975 as $3.1 million and $4.7 for the first ten months of 1976. This would extrapolate to about $5.6 million for 1976 and $8.7 million in total sales for the two years. How many systems would that be?

The average investment indicated on the Sweepstakes cards per system is $2032. That would give us about 4300 Altairs out there, hobby and commercial. Recent systems might be more expensive than older ones by virtue of higher prices, some buyers getting the floppy disk system with their Altair . . . etc., so older systems might be a little less expensive. Let's take 5000 Altairs as our fixed point of reference and figure that half of them are in hobby hands. As soft as the point of reference is, it's better than we've had before.

Using that reference and the percentages calculated from the cards we get approximately the following distribution of hobby systems:

Altair	2500
Imsai	1255
Southwest Tech	830

Kim	560
8080	1110
6800	788
8008	560
OSI	186
Sphere	155
Digital Group Z80	145
All others	2272

That gives us a total of a bit over 10,000 systems up and running or being built at this date. The statistics don't hold up quite as well with very small numbers, of course, since a sample of a bit over 1000 systems was used for the percentages. By way of checking the accuracy of the poll it was done in two batches of 500 and the percentages of the two counts were compared . . . the Altair came in exactly the same on both at 24.1%.

If any firm in the field has some better figures they certainly would be appreciated.

What about the 2272 miscellaneous systems? Well there were some significant numbers in there, like about 14 Sol systems being built (one running so far), 7 of the E&L MMD-1s, 14 of the RCA 1802s, five of the Intecolor 8001s not yet working, 9 F8 systems, 6502 Tims, 9900s, 6100s, Imp 16s, PACE, Mikes, etc. Multiply the numbers by about 10.3 for a rough idea of the number of each system out there. They're all under 2% of the market.

These figures put the total market more in the $40 million range (including 1975 sales), which seems a lot more in line with the size of the firms I've visited. This may explain why we haven't seen all that much growth of the manufacturing firms and why some have been up for sale . . . or sold. It also may explain why the expansion of computer stores seems to have slowed down a lot in recent months. What percentage of the sales are going through stores?

The hobby computer industry has apparently been growing as the result of energetic PR men and the purchases of a relative handful of enthusiasts. This explains why so many computer stores are having a very difficult time and why manufacturers are having to ship many of them on COD or prepaid terms.

There's no question in my mind that the field will eventually live up to its billing. The circulation of *Kilobaud* is proof of that. We're projecting a readership of 35,000 for this third issue of the magazine, and our Sweepstakes poll tells us that virtually every reader who doesn't have one is thinking seriously in terms of buying a system. Their own estimates of what they expect to spend is over $2000 on the average. This would put the buying power of the *Kilobaud* readership at a minimum of $50 million, since no more than 10,000 of them can have systems so far.

This would appear to give *Kilobaud* a particular advantage over some of the other magazines since the prospective customers seem to be heavily newcomers to the hobby . . . and *Kilobaud* is aimed right at the newcomers. This is reinforced by the hundreds of letters we are getting and the notes made on subscription forms.

One of the biggest problems facing the newcomer who would like to buy a system is trying to read the ads and the literature. I've made a positive pain of myself with the manufacturers pleading with them to write ads which even I could understand and to provide literature which would tell me enough to convince me that I should buy a particular system. I've been trying to get ad writing into the hands of people who remember what it is not to understand computers . . . so the ads would be written for people who are more than a bit hazy about the difference between a compiler and an interpreter . . . who would be hard up to explain a vectored interrupt . . . in other words, the majority of us.

By making sure that a fair percentage of *Kilobaud* is written at a level that even I can understand, I think we can educate a lot of newcomers to the point where they will not only understand how microcomputers work but will have to have their own systems and enjoy them. We'll try very hard not to succumb to the temptation to prove how smart we are by publishing a lot of scientific articles.

With the help of writers who are able to explain complicated ideas simply . . . reviews of most of the hardware on the market . . . lots of programs to make our systems more fun to use . . . and as much help with solving typical problems as we can publish, I think *Kilobaud* will appeal to a wide cross section of the computerists. Even more important, it will act as a bridge into computing for beginners who think they might be interested, but who are just starting out.

We have a splendid staff here at headquarters . . . not only to produce the magazine, but even more important to the whole field, to see that *Kilobaud* gets new subscribers from people not before into hobby computing. For instance we have a poster available which explains the magazine and has some subscription blanks on it. We'd like to get this poster on every bulletin board in the computer industry . . . manufacturers of computer hardware, accessories, software houses, etc. If you can help with this please drop a line to marketing manager Sherry Smythe, and she'll send you a poster.

We'd also like to get these posters on bulletin boards in schools with computer science departments . . . again, if you can help, please drop a line to Sherry.

If you run into any resistance on putting up a "commercial" poster, point out that people who have a personal interest in computers will be far more valuable to a firm . . . or do better in school. These are people who will be learning more about computers on their own . . . hobbyists always progress much faster than people who are just working at something for a living. *Kilobaud* will help people do better in school . . . and be far more valuable employees.

With a little help from all of the readers of *Kilobaud* we'll have 100,000 people into hobby computing within the next year and the manufacturers will be selling as much as they dream of now . . . and all dealers will be paying their bills. Nirvana.

HONEST EQUIPMENT REPORTS

Since it's unlikely that a lot of readers would be willing to pay $4 per issue for *Kilobaud*, the obvious answer is to run ads. This, of course, immediately sets up a conflict of interest situation between the editorial department of the magazine and the advertising department. It isn't as bad as you might think, once you understand all sides of the problem . . . and, having been an editor and publisher for 25 years, I'm pretty familiar with it.

On the one hand, if we publish anything even remotely critical of a product there is a good chance of making a life-long enemy. I tried that once in 1955 and learned a lesson. On the other hand, if we say everything out there is fabulous, the whole business is a waste of time and the product review is just a free ad.

Keep in mind two things about products. Firstly, few manufacturers spend all the time and money to get into production and advertise a product that is lousy. They may run into problems . . . bum parts . . . poor quality control . . . things like that. People are not yet perfect and it only takes one irresponsible person in a firm to make trouble . . . a test man . . . someone in the repair department . . . even the bookkeeper can raise holy hell for a firm until the boss gets feedback.

Secondly, manufacturers think of their products much as you and I do our children . . . they know there are some areas which can be improved, but woe to anyone outside who criticizes!

The solution I've worked out on this over the years has been to publish reviews of equipment which I think is good and to not publish them when I have problems. Since I get a chance (in the ham field) to check out just about everything that comes out, this isn't a bad row to hoe.

But suppose I wanted to be "honest" and let the chips fall where they may. In that case I'd have published a terrible review of a particular recent FM transceiver . . . the squelch is ridiculous on it . . . everything else is okay, but the squelch circuit is so bad that it ruins the whole radio. I would barely have had the review out

continued on page 22

kilobaud T.M.

PUBLISHER
Wayne Green
EDITOR
John Craig
PRODUCTION MANAGER
Ruth Brown
EDITORIAL ASSISTANTS
Kurt Schmidt
Peggy Sysyn
PRODUCTION DEPARTMENT
Manager:
Lynn Panciera-Fraser
Staff:
Craig Brown
Gayle Cabana
Robert Drew
Michael Murphy
Bob Sawyer
Noel R. Self
Robin M. Sloan
Jody Wright
TYPESETTING
Barbara J. Latti
Sandy White
PHOTOGRAPHY
Bill Heydolph
Ted Cluff
DRAFTING
Bill Morello
Lynn Malo
ASSOCIATE EDITORS
Don Alexander
Rich Force
John Molnar
MANAGER
Biff Mahoney
COMPTROLLER
Knud M. Keller
ASSISTANT COMPTROLLER
Marge Nielsen
MARKETING
Sherry Smythe
Karen McDonough
Lisa Joseph
ADVERTISING
Bill Edwards
Leslie Bailey
Nancy Cluff
Janet Ames
Barbara Hann
Lisa Healey
CIRCULATION
Dorothy Gibson
Nancy Chandler
Carol Dawdy
Janette Dyer
Florence Goldman
Pat Hogan
Lois Ireland
Judy Main
Theresa Toussaint
Marie-Anne Toussaint
PURCHASING
Susan Brumaghim
COMPUTER DATA CONTROL
Judy Waterman
Judy Brumaghim
Sherry Dean
Mary Jo Sponseller
COMPUTER ENGINEERING
C. Robert Leach
David E. Wilensky
Richard Dykema
PRINTING
Michael Potter
John Bianchi
William Cering
Brent Lawler
Gary Steinbach
INVENTORY CONTROL
Marshall Raymond
Larry Ames
Gary Slamin
PLANT MAINTENANCE
Bill Barry
Lorraine Pickering

Kilobaud is published monthly by 1001001, Inc., Peterborough NH 03458. Subscription rates in the U.S. and Canada are $15 for one year and $35 for three years. Outside the U.S. and Canada, please write for foreign rates. Application to mail at second class postage rate pending at Peterborough NH 03458 and additional mailing offices. Phone: 603-924-3873. INCLUDE OLD ADDRESS AND ZIP CODE WITH ADDRESS CHANGE NOTIFICATION.

20 Print "Editor's Remarks"
30 End
Run

John Craig

PROGRAMS ON RECORDS?

Maybe so. We're not really sure at this time whether it's such a hot idea or not. Therefore, we're going to check into it a little further. I had lunch with several interesting people the other day and we came up with a plan of attack. One of the gentlemen has some very tight connections in the recording industry, and we're going to have a record cut which will contain computer programs recorded at various speeds and formats. Hopefully, in the near future we'll have an article on the results.

We're not looking at this as a substitute for cassette recording but simply as another media to go with it. One of the nice features would be in the fact that *everyone* has a record player. Of course, *everyone* probably has a cassette recorder today, too. We're hoping the interfacing of the record player to the computer will involve no more than running a cable from the headphone or speaker output to the cassette interface already in the computer. Needless to say, the format of the data recorded on the record will have to be compatible with the interface.

What do you think?

TERMINOLOGY

Hey, don't be afraid to ask! If you're a newcomer to this game and you find yourself getting unfamiliar terms thrown your way, then by all means stop the thrower and ask him for a definition! Most of us are scared to death to show our ignorance but in a situation like this we're only perpetuating it. And it's not just the newcomer who needs to do it. We do have a problem with terminology in this field due to the fact different companies and schools use different terms to describe the same thing. As a result, it's not uncommon for us "old timers" to be hit with new terms every now and then. Invariably, when I stop someone during a conversation and ask them to explain a particular term it will turn out to be someting I was familiar with and had heard referred to by another name. A good example of this type of situation would be Dick Wilcox's series on Operating Systems (which, incidentally, will return next month). Dick uses the term "monitor" to describe the central, or main program, within the OS. I've always heard it referred to as the "executive" and my first reaction upon seeing "monitor" was that one of us must be wrong. Either term is correct . . . it just depends on where you've been and what your background is.

Don't be afraid to ask . . . and don't think the term you're using is the only one either, because it probably isn't.

THREE BIRDS WITH ONE STONE

I've got an idea to share with you on how you (or your club) can pick up some extra cash, introduce some new people to home computing, and also have one hell of a good time doing it!

Several weeks ago a friend of mine and I loaded up my station wagon with all of the electronics and computer junk we've accumulated over the years and headed for the local swap meet. We sold enough to pay for our gas! On the way home a brilliant idea was born. Since we needed electronic enthusiasts, fellow computer hobbyists, and hams to browse over the kind of goodies we were trying to sell, the only answer was to have a swap meet to draw them all together. So, we did! Not being one for thinking small, I called the event *The First Annual Central California Electronic/ Computer Hobbyists Swap Meet* (whew, what a mouthful!).

The more I thought about it, this seemed like a good opportunity to try and introduce some newcomers to computers. Therefore, I arranged to have several home systems up and running with some interesting programs. Unfortunately (as sometimes happens with the best laid plans), the demo systems didn't show up as expected and this particular aspect of the swap meet turned out to be somewhat of a flop. (We did manage to get two or three new people out to our next club meeting, which was okay. And, we did have several systems brought in by dealers which were up and running.) It's probably just as well the demo systems didn't show up because if I had it to do over I'd take a different approach. The ads I ran in the papers weren't really designed to stimulate the curiosity of the average layman . . . and, what the heck, if you're going to go to all this trouble, then why not try and get him to come down and take a look at a home system *and show him what it can do*. If you get him down there and he pops up with the question, "But what are you going to do with it?" then your display and demo programs have failed miserably. If you don't have the software then don't waste your time and his.

It isn't as much trouble to put on something like this as you might think. I had to get a temporary business and resale license, mail out the notices to all the stores in the area, run some ads in the newspapers, notify the hams in the area, and rent a building for a day.

Everyone had such a good time that there was a unanimous feeling the event should at least be semi-annual, rather than annual. Therefore, we're going to do it again in May . . . and this time we'll get *all* of California! Why heck, this last time we had six whole people come all the way from Los Angeles!

If your club would like to go into something like this in a big way (called a computerfest, rather than just a small-town swap meet) then let me suggest Wayne's December 16th *Kilobaud Newsletter* as a good source of information on how to go about it.

SPEAKING OF GET-TOGETHERS . . .

Don't forget the *First West Coast Computer Faire* coming up next month (April 15, 16, and 17th) in the Civic Auditorium in San Francisco's Civic Center. If you think last year's Trenton Computerfest was big, or the MITS Convention was large, or Personal Computing '76 in Atlantic City was huge, then you haven't seen big, large, or huge yet! It's coming in April and it'll be in San Francisco! And, what the heck, even if I'm wrong and the whole thing is a flop, you can't miss with a trip to magic San Francisco, right? As much as I love that city I really should be living there! (And you can bet that even a hint the Computer Faire will be a flop is a 100% put-on! It's going to be a ball!)

LOOKING BACK – MARCH 1975

And, speaking of San Francisco . . . it was on March 5th of 1975 that the Homebrew Computer Club was born in Gordon French's garage in Menlo Park, California. Gordon and Fred Moore got 22 people together for that first meeting and since then the Homebrew Club has grown into one of the largest in the country (the circulation of their newsletter is 1400 plus). Some of the people at that first meeting included Tom Pittman (Itty Bitty Computers), Bob Mullin (Mullin Computer Boards), Lee Felsenstein (designer of the Pennywhistle Modem and other goodies), Marty Spergel (M&R Interprises), and Bob Marsh (President of Processor Technology). Steve Dompier was the only person at that first meeting with an 8080-based machine up and running (an Imsai). He took a trip to Albuquerque and visited MITS shortly afterwards. At the second meeting of the club he reported that MITS had orders for almost 2000 units and the place was like a madhouse. Unfortunately, they were only building 256-word memories to go with their systems. Bob Marsh walked up to Steve during that meeting and asked, "What do you suppose it takes to put together a larger memory board?" And, from that question Processor Technology was born within a couple of weeks. Gordon French later joined Processor Tech and was the Project Manager for the SOL (he is currently with Imsai).

It was also in this third month of 1975 that *The Computer Hobbyist* (Steve Stallings and Hal Chamberlin in Cary, North Carolina) came out with their cassette interface which, for a time, looked as though it might become *the* interface for the home system. The interface utilizes pulse modulation encoding, was designed to control two cassette units, and receives high praise in the reliability department from those who use it. As a matter of fact, the *TCH* interface is available in a kit from Electronic Discount Sales (Mesa AZ) for $28.50. Data transfer rates up to 1000 baud are possible with this unit. My gosh, that's a KILOBAUD, isn't it?

MISCELLANEOUS

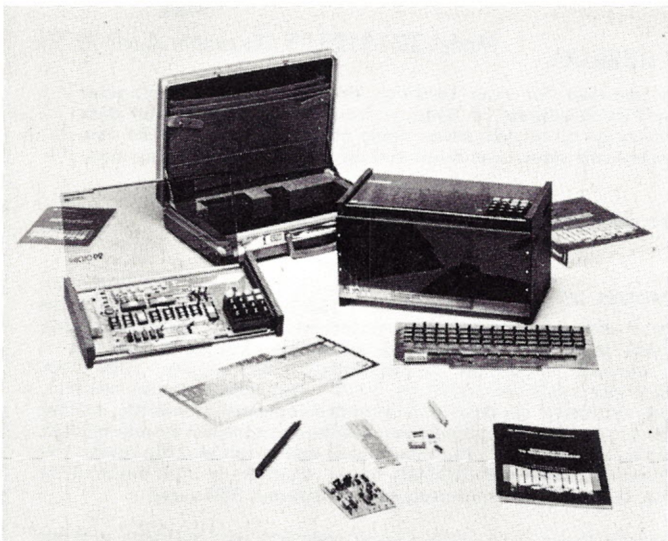
You've heard about these new 12 volt automobile batteries that have lifetime guarantees and don't require any water? Sounds like it would be ideal for a standby power source so the contents of memory wouldn't be lost each time we power down (it's something you get used to, and like . . . I know, I've got core memory in one of my systems). I'd like to see a construction article which described building a charger and low-battery indicator (what could be simpler?). I've heard these batteries aren't really any improvement over what we've been buying in years past, it's just that they figure you're going to trade that car in within the next few years and the warranty doesn't go with it. Therefore, it might be a good idea to make sure the warranty will cover this application. Either that, or have it installed in your car . . . and bring it back that way!

My thanks to Jeff Fint at the local school (Cabrillo) for the fine job he did in putting together my Imsai . . . because I was too busy trying to put together magazines.

NEWS OF THE INDUSTRY

LOW COST 8K MICRO COMPUTER MEMORY SYSTEM

Electronic Product Associates, Inc., has announced the availability of a new, low-cost 8K x 8 bit memory system for microcomputer use. The memory uses 91L02A/21L02B-1 500 ns static memory chips. It features complete, switch selectable address decoding and fully buffered address and data lines. The memory comes completely assembled and tested on a 8" x 15" (20½cm x 38cm) board for $320.00 F.O.B. San Diego. Power required is 5 volts at 1.5 amps. An 86 pin edge connector uses the Motorola Exorciser bus pinouts and plugs directly into the EPA Micro-68 computer expansion cabinet. Delivery is from stock. For additional information contact: Electronic Product Associates, Inc., 1157 Vega Street, San Diego CA 92110, (714) 276-8911.



PROM PROGRAMMER FROM MITS

MITS' new 88-PROM Programmer, designed to work with the Altair 8800 systems, will program the standard 1702A (256 byte) erasable PROMs in less than three minutes.

The unit consists of a separate chassis (10.6" x 4.2" x 11") with a 24 pin zero insertion force socket. This is connected to the 8800 through its own interface card which plugs into the Altair bus. The programmer has a self-contained power supply, and the interface card requires less than 500 mA from the +8 volt bus in the 8800.

The programmer requires an 8800 (a or b) computer with an 88-PMC (PROM Memory Card), an SIO or 2SIO (Serial Input/Output Card) and some type of terminal device. The 88-PMC is necessary because the software driver for the programmer will be supplied on PROM.

The programmer looks like an addressable output port with two channels (even and odd). The even address channel outputs to a "control latch" in the programmer. The odd address channel outputs to either an "address latch" or a "data latch" depending on the state of the fourth bit in the "control latch."

The 8800 controls all the programmer timing through its software driver. The PROM is programmed by doing a 256 byte block transfer from the 8800 memory (RAM or PROM) to the programmer.

Programmed PROMs are checked by placing the PROM in a standard PMC and running the check routine which is part of the supplied software.

The Altair 88-PPC Programmer is available only as an assembled unit ($456).

For additional information contact Mits at 2450 Alamo S.E., Albuquerque NM 87106.

THE PCM-12A

The newest PDP-8 software-compatible microprocessor product is now available from PC/M, Inc. The late addition to their line is the PCM-12A, a 12-bit microcomputer designed around the Intersil IM6100 microprocessor. Like its predecessor, the popular PCM-12, the 12A is fully compatible with DEC's software for the PDP-8/E minicomputer, and is an expandable bus-organized system with capacity for up to 32K words of memory.

Several features added to the PCM-12 resulted in the 12A designation. Major improvements to the Central Processor module have produced "quieter" signal distribution on the 15-position backplane bus. The CPU module also includes a built-in crystal-controlled Baud-rate generator which services all asynchronous I/O device interfaces on the bus. The control-panel design has been expanded to include an absolute loader which will directly bootstrap a binary-format tape into any field of memory at the press of a front-panel switch. Front-panel nomenclature has been improved for stronger PDP-8 compatibility. Cabinetry and documentation have also been "beefed-up", and a floppy-disk mass-storage capability has been added to the system.

The machine is available in both kit and assembled form. Either way, the microcomputer is fully compatible with nearly all software written for DEC's PDP-8 family. With addition of the floppy-disk for mass storage, the PCM-12A is even capable of running DEC's OS-8 and Intersil's IFDOS operating systems. Available interfaces include serial and parallel I/O, high-speed reader/punch (all DEC-compatible), and an audio-cassette recorder interface. Memory modules include NMOS and CMOS RAM and a flexible EPROM module. Priced in kit form as low as $799.00, with CPU, control-panel, 1K static RAM memory, cabinet and power supply. For more information, contact Pacific Cyber/Metrix, Inc. (PC/M), 180 Thorup Lane, San Ramon CA 94583, (415) 837-5400.

RANDOM ACCESS MEMORY SYSTEM

The Smoke Signal Broadcasting M-16 is a 16,384 x 8 bit semiconductor random access memory system, completely contained on a single printed circuit card assembly. It is plug compatible with the Southwest Technical Products Corporation 6800 microcomputer and allows expansion to 32K without any modifications. Expansion to 48K is possible with a simple modification to the SWTPC 6800.

The M-16 uses AMD 9141ADC static memory chips and is fully buffered. The system requires just one 8 volt power supply and draws approximately 1.8 amps (which is less than half the power requirement of a similar size system constructed with 2102's).

Measuring 5½" x 9", the M-16 is the same size as the SWTPC 6800 CPU and memory boards. It combines speed, sufficient to run the SWTPC 6800 at full clock speed, with very compact design at a surprisingly low cost. This is the first of a number of products designed by Smoke Signal Broadcasting to make the SWTPC 6800 a truly complete, low-cost computer system.

The M-16 is priced at $595 and delivery is from stock.

For further details, contact: Ric Hammond, Smoke Signal Broadcasting, P.O. Box 2017, Hollywood CA 90028, (213) 462-5652.

INTERSIL INTRODUCES NEW FREQUENCY COUNTER TIME-BASE

Intersil has broadened its line of timing microcircuits through the addition of the ICM7207A, a new frequency counter timebase. Used together with a 5.24288 MHz crystal and a 7 digit unit counter such as Intersil's ICM7208, the new circuit becomes a complete timer-frequency counter.

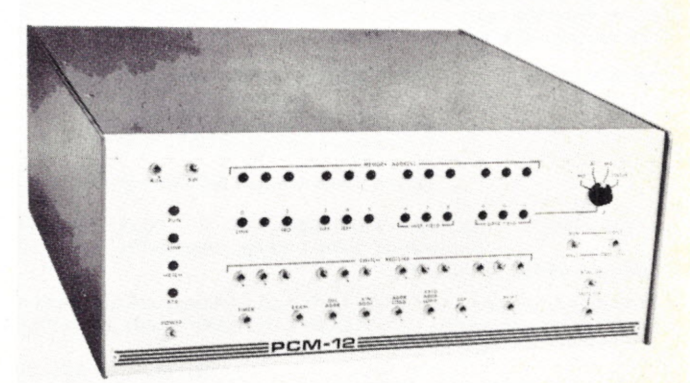
The new circuit is pin-for-pin compatible with Intersil's ICM7207, however it has 0.1 and 1 second count enable window output.

When used with the ICM7208, the circuit's four outputs provide the gating signals for the count window, store function, reset function and multiplex frequency reference. The 1 second count enable makes it possible to obtain 7 significant digits when measuring frequencies over 1 MHz with the least significant digit reading in Hz.

The ICM7207A will take crystals from 1 to 10 MHz, providing outputs at crystal frequency, and at $\div 2^{12}$, $\div 2^{20}$ or $\div (2^{20} \times 10)$ divider stages.

The new circuit has a stable HF oscillator which dissipates less than 5 mW at 5 volts.

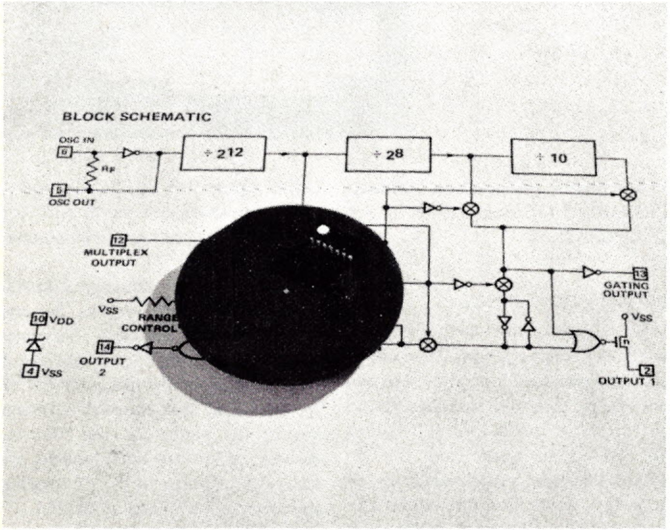
According to Intersil, the new cir-



cuit will be quite useful for applications requiring a system timebase, oscilloscope calibration generator, marker generator strobe, or frequency counter controller.

The circuit is packaged in a 14 pin DIP. Dice are also available.

For more information, contact: Intersil, Incorporated, 10900 North Tantau Avenue, Cupertino CA 95014, (408) 996-5000.



MEK 6800D2 EVALUATION KIT II

MEK6800D2 provides a useful and expandable tool for those who wish to develop systems with the MC6800 microprocessor without investing in expensive terminals. All parts needed to complete the system and get on the air are provided in the kit with the exception of the power supply. In addition to the expansion available on the basic microcomputer module, additional RAM, ROM, and I/O parts can be accommodated at a later date to implement more complex systems. Machine language programs can be entered through the system keyboard or via a built-in audio cassette interface system. Hexadecimal LED displays are provided for monitoring data and address information. A crystal controlled clock generator is used to eliminate timing adjustments.

FEATURES:

* Easily expandable
* Wire-wrap capability
* Parallel and serial interface capability
* 16 I/O lines/4 control lines
* Single 5 V supply required
* J-Bug monitor
  - Trace one instruction
  - Set up to five breakpoints
  - Examine and change registers
  - Punch and load cassette to memory
  - Examine and change memory
  - Execute User Programs

For more information contact: Motorola Semiconductor Products Inc., 3501 Ed Bluestein Boulevard, Austin TX 78721, (512) 928-2600.

PLEASE, THINK SMALL

ATTENTION KIM-1 OWNERS. Now that you have your KIM-1 attached to the power supply and have successfully added 2 + 3 and gotten 5, would you like something else to do? Would you like to use the KIM-1 as:

a TIMER accurate to a millisecond

a CLOCK displaying hours, minutes and seconds

an ADDING MACHINE with six digit add/subtract for the old checkbook

a DECIMAL-HEX/HEX-DECIMAL Converter

a DRUNK TEST

a simple GUESS-THE-NUMBER game for the kiddies

the MASTERMIND game for you

the SHOOTING STARS puzzle

a series of REACTION TIME tests

a MOVING MESSAGES DISPLAY with Alphabetic Characters

plus other demos, tests and games?

Would you appreciate having all of these capabilities in an integrated software package that includes a "high level language" which will let you create your own programs???

MicroCosmos announces *PLEASE*, a package which contains all of the above features and runs on the basic KIM-1 – no additional memory, TTY, or peripherals required. *PLEASE* is distributed as a CASSETTE TAPE and includes complete SOURCE LISTINGS, full OPERATING INSTRUCTIONS, and instructions for writing your own programs in *PLEASE*. The total cost: $10.00. MicroCosmos, 210 Daniel Webster Highway, So., So. Nashua NH 03060.

ADD A CALCULATOR CHIP TO YOUR MICRO!

At last! Now you can have full floating point calculations without using a lot of memory. With the RSG Electronics mathematical function unit board, you get not only the standard add, subtract, multiply and divide, but a full 40 function floating point calculator array. The RSG MFU puts at your command many features not found anywhere else including:

Complete compatibility with any computer having one TTL level I/O port.

Straightforward programming using I/O commands.

Speed equivalent to, or better than, software when performing complex functions.

Special functions may be programmed into the MFU and allowed to operate independent of further processor control.

Built-in overflow and error detection.

Highest quality board available with clear and complete instructions – with through hole plating and gold-plated fingers for a reliable trouble-free product.

In stock now and available from: RSG Electronics, P.O. Box 13, Santa Margarita CA 93453.

NEW RCA COSMAC MANUAL FOR THE DESIGN ENGINEER . . . and the computer hobbyists?

Written for electronics engineers having only a limited familiarity with computers and computer programming, this new *User Manual for the RCA CDP1802 COSMAC Microprocessor*, MPM-201A, published by RCA Solid State, guides the reader through the microprocessor architecture and introduces a set of comprehensive, easy-to-use programming instructions.

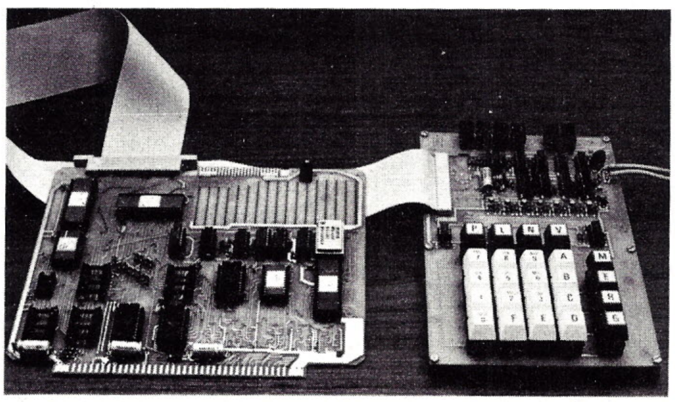
This manual is a detailed guide to the application of the COSMAC microprocessor CDP1802, a one-chip CMOS 8-bit register-oriented central processing unit suitable for a wide range of stored-program computer systems, both special and general purpose.

Examples illustrate the operation and usage of each of the 91 instructions and aid the design engineer in developing simpler and more powerful products utilizing the wide range of microprocessor capabilities.

For system designers, this User Manual illustrates practical methods of adding external memory and control circuits. Detailed examples show the use of I/O instructions and I/O interface lines, including DMA and interrupt inputs, external flag inputs, command lines, processor state indicators, and external timing pulses.

The manual covers various programming techniques, with examples, as well as more advanced topics such as the use of subroutines, interrupt service, and RAM and register allocation.

Copies of the 115-page, 8-½ x 11-inch, *User Manual for the RCA CDP1802 COSMAC Microprocessor*, MPM-201A, containing 125 figures, may be obtained at $5.00 a copy from RCA Solid State Division, Box 3200, Somerville NJ 08876 or from RCA Solid State distributors.



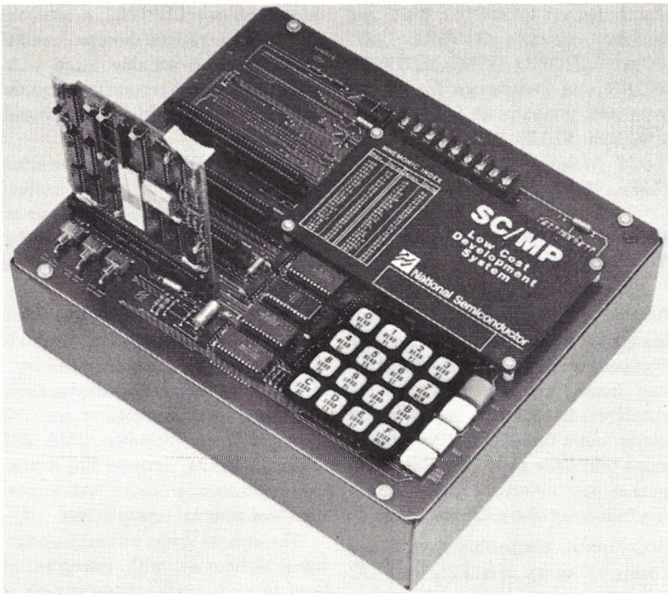
SC/MP MICROPROCESSORS ADD LOW COST DEVELOPMENT SYSTEM

Hardware and software designs that involve the "SC/MP" 8 bit microprocessor can be easily developed and quickly tested with a new "Low-Cost Development System" (LCDS) now available from National Semiconductor in the form of a self-contained microcomputer.

The Low-Cost Development System is a simple controller configured with a SC/MP CPU (central processor unit) card plugged into one of four sockets in a card bus on a 10 x 12 inch motherboard. Along with the CPU card, the motherboard also contains a 16-key, dual function hexadecimal keyboard, four keys that control function, three control switches and a 6 digit hexadecimal LED display.

With the basic LCDS configuration alone, control logic, scratchpad memory and ROM-based firmware on the motherboard allow the user to alter the SC/MP registers and memory locaters, run SC/MP programs in continuous or single instruction mode, and even operate with an optional Teletype using SC/MP DEBUG.

The SC/MP LCDS and supporting cards may be ordered directly from National Semiconductor, or National's local franchised distributors. Pricing of single units of the LCDS (ISP-88/30) is $499.00, with the 2K x 8 RAM card (ISP-8C/002) priced at $160.00 each. For further information contact marketing manager: Hashmukh Patel, (408) 737-5173.



TARBELL 1010

The Tarbell Electronics Model 1010 Prototype Board, for the ALTAIR and IMSAI computers, accepts up to 33 14-pin ICs or a mixture of 40-pin, 24-pin, 18-pin, 16-pin, and 14-pin ICs. The board is mainly oriented toward soldering point to point, but wire-wrap may also be used.

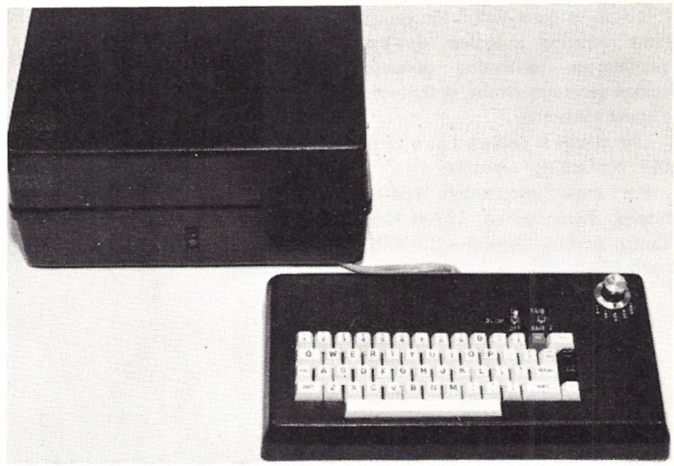
There are three rows for ICs. The ICs or sockets are inserted from the top, then wires are soldered to the adjacent tabs. The tabs are such that even with 40-pin ICs, there are still two holes left over on each pin for wires. A place for a 5 volt regulator is also provided. Edge pins are gold-plated. The price for the Tarbell Model 1010 is $28.

For further information, contact: Tarbell Electronics, 20620 S. Leapwood Avenue, Suite P, Carson CA 90746, (213) 538-4251.

ENCLOSURE KIT FOR THE CT-1024 TERMINAL

PARSEC ELECTRONICS is now marketing an enclosure kit for the SWTPC CT-1024 Terminal System. A two enclosure design is used to give the terminal suitable protection while retaining the flexibility of a movable keyboard for the user. The two cases are formed from strong, resilient ABS plastic and are color coordinated to blend in with the SWTPC 6800 computer system. The keyboard enclosure has the cutout and mounting pads for the KBD-5 keyboard and a fully enclosed bottom.

The main case features mounting pads for the CT-1024 main board, the GT-61 graphics board, the AC-30 cassette interface board and room for their power supplies. The kit includes mounting hardware for all circuit boards, a heat sink for the power supplies and rubber feet for both enclosures.



For more information contact: PARSEC, P.O. Box A82327, San Diego CA 92138.

PROBIT LOGIC PROBE

The PROBIT tests both TTL/DTL and CMOS logic systems. It can detect pulses down to 35 nanoseconds and accurately check for valid and abnormal logic levels. No loading is added to the circuit under test due to PROBIT's high input impedance of over 1kΩ. Protection against overvoltage inputs is provided along with reverse voltage protection to 100V. Seven segment display indicates "H" for logic high, "L" for logic low, and a "period" for pulse detection. Display is blank for abnormal logic levels. Convenient "Micro-Hooks" are used for test connections, and the probe tip is detachable.

The new PROBIT sells for $34.95 from Control and Information Systems, 10 Spring Valley Village, Richardson TX 75080.

88-MUX 24-CHANNEL MULTIPLEXER

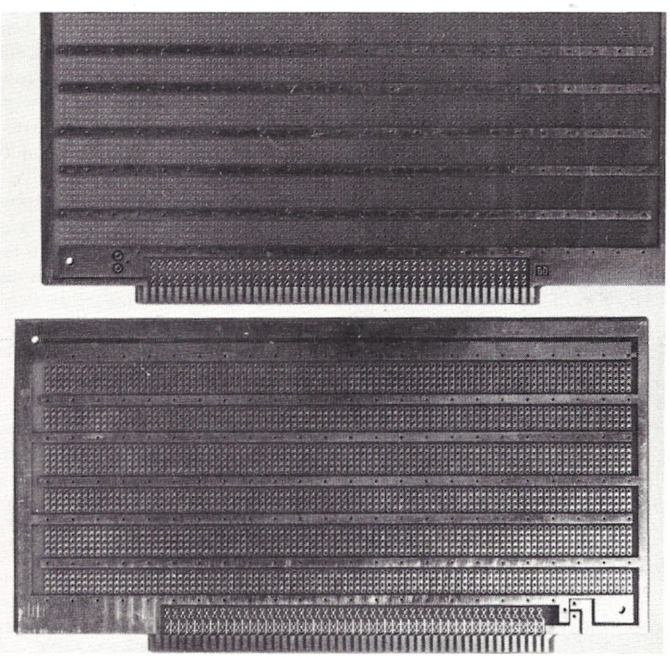
MITS, Inc., is pleased to announce the introduction of the 88-Mux (a companion card to the 88-Analog/Digital Converter), which will expand the input capacity of the 88-ADC for applications requiring a large number of analog inputs.

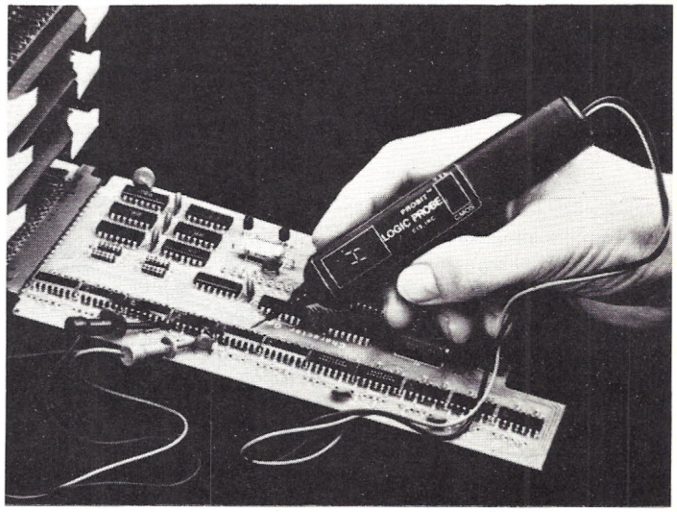
The 88-ADC is actually a stand-alone card for many systems, because it contains an on-board 8-channel multiplexer. But for a majority of system layouts, the real potential of the ADC and Mux conversion system lies in the ability of the Mux to process more than eight signals. By using four 88-Mux cards, it's possible to process up to 96 analog signals!

Another advantage of using the new 88-Mux is the optional differential input on each channel. As a bonus, the gain and scale factoring of each channel can be set independently, giving extreme flexibility in system design. Input filtering can also be added to provide the desired roll-off characteristics.

An appropriate interface cable is provided with each ADC and Mux pair. (For more than one 88-Mux per system, a cable is required for each Mux card.)

The 88-Mux will be available within 60 days of order placement at a cost of $319 (assembled only). For further information contact: Mits, Inc., 2450 Alamo S.E., Albuquerque NM 87106.





DIODES/ZENERS

1N914	100v	10mA	.05
1N4004	400v	1A	.08
1N4005	600v	1A	.08
1N4007	1000v	1A	.15
1N4148	75v	10mA	.03
1N753A	6.2v	z	.25
1N758A	10v	z	.25
1N759A	12v	z	.25
1N4733	5.1v	z	.25
1N5243	13v	z	.25
1N5244B	14v	z	.25
1N5245B	15v	z	.25

SOCKETS/BRIDGES

8-pin	pcb	.25	ww	.45
14-pin	pcb	.25	ww	.40
16-pin	pcb	.25	ww	.40
18-pin	pcb	.25	ww	.75
22-pin	pcb	.45	ww	.75
24-pin	pcb	.35	ww	1.25
28-pin	pcb	.35	ww	1.45
40-pin	pcb	.50	ww	1.95

Molex pins .01 To-3 Sockets .25

2 Amp Bridge 100-prv 1.20

25 Amp Bridge 200-prv 2.50

TRANSISTORS, LEDS, etc.

2N2222	NPN			.10
2N2907	PNP			.15
2N3740	PNP	1A	60v	.25
2N3906	PNP			.10
2N3055	NPN	15A	60v	.50

LED Green, Red, Clear .15

D.L. 747 7 seg 5/8" high 1.95

XAN72 7 seg com-anode 1.50

FND 359 Red 7 seg com-cathode 1.00

C MOS

4000	.20
4001	.20
4002	.25
4004	4.95
4006	1.20
4007	.40
4008	1.20
4009	.25
4010	.45
4011	.20
4012	.25
4013	.40
4014	1.10
4015	.95
4016	.35
4017	1.10
4018	1.10
4019	.70
4020	.85
4021	1.35
4022	1.15
4023	.25
4024	.95
4025	.35
4026	1.95
4027	.50
4028	.95
4030	.45
4033	1.95
4034	2.45
4035	1.25
4040	1.35
4042	.95
4043	1.25
4044	.95
4046	1.50
4049	.80
4050	.70
4066	1.35
4069	.40
4071	.35
4082	.45

— T T L —

7400	.15
7401	.15
7402	.20
7403	.25
7404	.15
7405	.25
7406	.45
7407	.55
7408	.25
7409	.15
7410	.15
7411	.25
7412	.30
7413	.65
7414	1.10
7416	.25
7417	.50
7420	.15
7426	.40
7427	.45
7430	.15
7432	.45
7437	.45
7438	.35
7440	.25
7441	1.15
7442	.65
7443	.95
7444	.95
7445	.95
7446	.95
7447	.95
7448	1.20
7450	.25
7451	.25
7453	.25
7454	.25
7460	.40
7470	.45
7472	.45
7473	.35
7474	.40
7475	.45
7476	.20
7480	.65
7483	1.00
7485	1.05
7486	.40
7489	2.50
7490	.40
7491	1.15
7492	.95
7493	.45
7494	1.25
7495	.85
7496	.95
74100	1.85
74107	.45
74121	.40
74122	.55
74123	.55
74125	.45
74132	1.35
74141	1.30
74150	1.00
74151	.95
74153	.95
74154	.75
74156	1.15
74157	.75
74161	1.25
74163	1.25
74164	.95
74165	1.50
74166	1.35
74175	.95
74176	1.25
74180	.85
74181	3.25
74182	.95
74190	1.75
74192	1.65
74193	.85
74194	1.45
74195	.95
74196	1.50
74197	1.25
74198	2.35
74367	.85
75108A	.35
75110	.35
75491	.50
75492	.50
74H00	.25
74H01	.25
74H04	.25
74H05	.25
74H15	.30
74H20	.30
74H22	.40
74H30	.25
74H40	.25
74H52	.15
74H53J	.25
74H55	.25
74H72	.55
74H101	.75
74H103	.75
74H106	.95
74L00	.35
74L02	.35
74L03	.30
74L10	.35
74L30	.45
74L47	1.95
74L75	.55
74S00	.55
74S02	.55
74S03	.50
74S10	.45
74S11	.45
74S20	.50
74S40	.30
74S51	.45
74S64	.30
74S74	.50
74S112	1.50
74S133	.45
74S140	.75
74S151A	.45
74S153	.45
74S158	.45
74S194	1.50
74S257 (8123)	.25
74LS00	.45
74LS01	.45
74LS02	.45
74LS04	.55
74LS08	.45
74LS09	.45
74LS10	.45
74LS11	.45
74LS20	.50
74LS21	.25
74LS22	.25
74LS32	.55
74LS37	.40
74LS40	.55
74LS42	1.75
74LS52	1.45
74LS74	.95
74LS90	1.30
74LS93	1.00
74LS107	.95
74LS153	1.20
74LS157	.85
74LS164	1.90
74LS368	.70

9000 SERIES

9301	1.00
9309	.45
9602	1.50

MEMORY, CLOCKS

74S188 (8223)	3.00
8080	26.50
MM1702A	10.50
MM5314	3.50
MM5316	3.95
2102-1	1.75
2102-L1	1.95
TR 1602A	6.95

LINEARS, REGULATORS, etc.

MCT2		.95
8038		3.95
LM201AH		.75
LM301AH		.25
LM308AH		1.00
LM309H		.65
LM309K		.90
LM310		1.15
LM311		1.35
LM320K5		1.65
LM320K12		1.65
LM320T12		1.65
LM320T15		1.65
LM339		1.65
7805 (340T-5)		1.00
LM340T-12		1.25
LM340T-15		1.25
LM340T-18		1.65
LM340T-24		1.25
LM340K-12		2.15
LM340K-15		1.65
LM340K-18		1.65
LM340K-24		1.25
LM373		1.95
LM380		.95
LM709		.30
LM711		.45
LM723		.45
LM725		1.95
LM739		1.50
LM741	8-14	.25
LM747		1.10
LM1307		1.25
LM1458		.95
LM3900		.65
LM75451		.65
NE555		.50
NE556		1.10
NE565		.95
NE566		1.75
NE567		1.35
SN72720		.35
SN72820		.35

INTEGRATED CIRCUITS UNLIMITED

7889 Clairemont Mesa Blvd. • San Diego, CA 92111 • (714) 278-4394

All orders shipped prepaid — No minimum

Open accounts invited — COD orders accepted

Discounts available at OEM Quantities

California Residents add 6% Sales Tax

24 Hour Phone (714) 278-4394 — **MasterCharge / BankAmericard**

Letters to the Editor

MAILING LIST

Here are some random thoughts. Pleased to see the emergence of the *Kilobaud* newsletter. We have an urgent need for some way of contacting potential dealers for our UC1800 and future items. I would like to see an *independent* mailing list of dealers. I would guess you have such a thing. Why not publish it?

How about a chart (like EDN), but showing dealers with the types, categories, and (brands?) of equipment they carry. Perhaps, their one most burning need also. Done on a quarterly questionnaire basis, it should yield a computer full of statistics and perhaps good trend info.

We find a tremendous interest shown by the industrial segment in the UC1800. That's where the sales are too. I'm not sure I really understand where the "hobby market" is or what it is. The potential hobbyists I have had personal contact with seem to have the following outlook:

1. They know they want a computer.
2. They don't know what they will do with it.
3. They want it to perform a significant function.
4. They would like an IBM 360.
5. They feel $295.00 is too much money to spend.

Surely, I must have a distorted view of this market!

W. J. Haberhern
Cape Canaveral FL

Bill, you have the hobbyist pegged just right . . . congratulations! Re the dealer mailing list . . . that costs us a lot of money to keep up to date so it seems reasonable to pass that cost along and charge $50 per printout of our list of dealers, clubs and manufacturers. We will be asking dealers to let us know what lines they carry and pass this info along. We have a problem . . . there are a lot of things we'd like to do and not enough people to do them . . . so I end up sitting here writing a newsletter . . . Wayne.

QUESTIONS

I have several questions on microcomputers (about 1000) but there is nowhere I can find the answers here in Belgium. Perhaps, you can give me a couple of references or, perhaps, someone at 73 knows the answers. What is the difference between Imsai/Altair bussing and other arrangements? What is the difference between dynamic and static support chips, RAMs, ROMs, etc? I am considering buying a small uP system, the SC/MP from National, but don't know if I could expand this system to more than just the basic parts. Is the bussing system compatible with the other popular uP systems?

As you can see, I don't know much about micros. I guess I'm not alone, though I do want to learn. Keep up the fine tradition of 73 magazine and I'll be looking forward to *Kilobaud*.

SFC John W. Daugherty
Shape APO NY

Hopefully, the letter from Bob Leach (our Computer Engineer) cleared up some of those questions, John, but I'm sure there will be many more cropping up in the future. Hopefully (again), Kilobaud *will take care of those for you. – John*

UPDATE

Re my article "A 6800 Single Stepper" in last month's issue of *Kilobaud*, you might be interested in knowing that IC1 in Fig. 1 is a 77400 Quad NAND gate and the resistor is 1k Ohm, ¼ Watt.

Mark Borgerson
Corvallis OR

ANSWERS, ANYONE?

I am most interested in obtaining more information on hobby computers. Unfortunately I live in a rural area and to the best of my knowledge there is no one, no clubs and of course no sales outlet for hobby computers.

Information required is,

a) What can a hobby computer do and how can it be used in the household?

b) What interface devices are necessary, and which are recommended?

c) What features to look for (and which to avoid) when purchasing?

d) A list of good starter kits which can be expanded at a later date.

e) Recommended reading for the beginner?

Plus any other information you may think is necessary for the beginner.

Good luck with your new magazine and as you can see I have signed up as a Charter Subscriber.

Wm. A. Rendall Jr
RR #2, Duncan
British Columbia
Canada, V9L 1N9

These are all subjects which will be covered in detail in Kilobaud. *– John*

SHORT RESPONSE TIME

I would also like to thank you for the quick response on the RTTY Evolution article for the 73 I/O section. I just got the proofs Friday and will send them in tomorrow. With response times as short as 2 months, I think I will be writing many more articles for you. I can't live with the 8 to 12 month lead time of the Brand X magazine.

Bob Brehm
Menlo Park CA

Thanks, Bob. If my response time was over one month *then it was too long. – John.*

DIGITAL TECHNOLOGY

For some months I've been pondering an article or series of articles, introducing and surveying the present state of digital techniques applied to music recording and music signal processing. After attending the Audio Engineering Society Convention in New York, last month, I am sure that now is the time: digital technology is popping up everywhere in a usually strictly analog domain. Examples?

Dr. Tom Stockham (of U. of Utah) *demonstrated* a practical digital sound recorder which sounded as good, if not better, in some ways than traditional professional recorders. He is currently on leave of absence from the University to develop an all digital recording studio. Barry Blesser (M.I.T.) delivered a paper and demonstrated an all digital reverberation synthesizer based on a dedicated microprocessor. The device will be for sale next year and might revolutionize professional recording studio echo techniques. Several Automation Systems were displayed for the control of volume levels, panning, echo, and equalization in the mixdown of 24-track master tapes to final two-track master tape or disc form. One is based on a "naked mini", another uses an inexpensive 8080 system with extensive software and a digital cassette slaved to the motion of the 24 track tape machine, a third offers a dedicated version of the ISC Intecolor 8001, with amazing graphics and operator interactions possible.

There were at least three "search to cue" systems shown, all based on simple microprocessor circuits which control the fast-forward and rewind of the master recorder to search out spots on the master tape that had previously been introduced into RAM. Thus, the engineer no longer needs to rewind by ear after a take: he simply punches in the place on the reel to which the machine must go and the "computer" does the rest. A variation on this type of machine control uses a time code recorded on the tape and can be used to synchronize two or more recorders (or an audio and a video recorder) or can even perform rudimentary editing.

All are based on microprocessor systems easily understood by the computer hobbyist, and could spark dozens of ideas for home music central control. This field is just starting to jump and I feel hobbyists will be fascinated to hear what is being done. More importantly, the field is so new that the hobbyist could easily involve himself and possibly make important advances if he had some basic information such as: who's doing what with which chips and what applications are crying out for experimentation?

I hope this has jogged your imagination. If not, I've attached a rough list of a few potential topics I think might interest the computer hobbyist. *Kilobaud* seems particularly apt (aside from the fact that I'm a fan of 73 and Wayne Green) since you and Wayne have espoused *basic* instructional articles. I feel that a "What's happening here?" sort of survey will answer a lot of questions and arouse interest in this branch of small computer applications. There is much basic work to be done, lots of fun to be had, and little or no information in popular magazines as yet.

Tom Scott
Mill Valley CA

Needless to say, I didn't waste any time getting back with Tom regarding his ideas. Keep your eyes open for his articles in upcoming issues . . . he's got some interesting things to share with us. – John.

FLOWCHARTS

Another tip: Could you please print clear and complete flowcharts of the problems you want to solve (or let be solved) in your articles? Flowcharts seem to be as universal as "BASIC" and can be used to reprogram to other object codes. I think that those program listings of 8080 (or other) problem-solving programs are of little use to 6800 (6501 ..., Z80, 8008 ...) users. A flowchart would help to rewrite a program to one's own system. I do use a MOS Technology KIM-1 system (6502, Display, KB, Tape and TTY interface ...).

Rupert Mohr
Duisburg 14 Germany

HOBBYISTS AND INDUSTRIAL USERS – COMMON PROBLEMS

My own interests are mainly outside the hobby area, particularly in engineering education and microcomputer system development. However, I hope to continue a fairly close association with the hobbyists. As you know, the hobbyists in San Diego hold their meetings at Grossmont College. And frankly, we have learned a tremendous amount from them as to what is available from whom, how things really work, and what products are worth buying. We encourage hobbyists as students and try to aim our courses to handle people with a wide variety of backgrounds. Hobbyists have a tremendous amount to offer in the educational area, particularly for those schools which have limited facilities and staff to cope with this rapidly changing technology. The benefits can flow both ways since schools have facilities and equipment that hobbyists may otherwise find it very difficult to obtain. I hope that both hobbyists and educators (they may often be the same people!) will search for ways to work together and that hobbyists will not assume that those teachers who look down on the nonacademic world are typical.

Although there are differences between the hobbyists and industrial users and these differences will probably grow larger, there are many common problems:

(1) How do we take a problem and formulate it in a way that can be implemented on a computer? This encompasses problem definition and perhaps some other aspects as well.

(2) Once the problem has been defined, how do we design the software and hardware to solve the problem? This involves both program and interface design. Program design techniques may include flowcharting, structured programming, top-down design, etc.

(3) How do we actually implement the software and hardware? The software method is coding in a computer language. The hardware methods include breadboarding, soldering, wire-wrap, etc. Note that this stage, which initially might seem to be the most difficult, is actually the simplest once you learn the basic methods. The hardest part is deciding what to do. Hobbyists with some software background might be interested in a useful little book – Kernighan and Plauger, *The Elements of Programming Style*, McGraw-Hill, New York, 1974. The book costs $5 and is full of hints on how to program in a reasonable manner. Another useful book is Weller, *Assembly Level Programming for Small Computers*, Lexington Books, Lexington, MA, 1975.

(4) How do we get the system to work the way we want? This is, of course, debugging or verification. If we write the program properly (and implement the hardware carefully), we can make this stage much simpler.

(5) How do we make sure that the overall system really works properly? This step is testing or validation. I should note that, if we are to have software libraries (and perhaps interface libraries as well), we must have some standard way of verifying and validating programs and hardware.

(6) How do we document the system? Surely some standards here are essential if we are to exchange programs or figure out what someone else's program (or perhaps even our *own) does.*

All of these questions must be settled by both hobbyists and industrial users. Some of the equipment and methods may be different, but the basic problems are the same. These are the subjects that I'm concerned with, both in teaching and in system development. I plan to discuss them in some articles later for *73* or *Kilobaud* magazine.

Lance A. Leventhal
Solana Beach CA

DON'T FORGET THE NOVICE

Here it is Christmas Eve and guess what the mailman arrived with? My first issue of *Kilobaud*. Well, most of the day was spent with the kids but I did manage to sneak in a few hours reading. May I say your first issue was great!!!

One comment about future issues of your magazine. Although I enjoy advanced ideas in hardware and software, I still fit in with the Novice. I have worked with the computer several years, but I'm never "too old to learn." Don't forget about newcomers and people like me who don't have "kilobaud" transmission rates and "Megabyte" memory.

Also, bravo to Dick Wilcox letter! I agree 100%. I must admit though, I wish I could make this viewpoint from a professional view. Not just from the receiving end.

One problem though with your magazine: page 39. How dare you entice me into cutting up my magazine with an entry blank for a sweepstake? Even excellent prizes will not make me do so, but I hope you have the kindness to overlook my boldness and enter my name in your sweepstakes anyway. Please!!!

Wilbert M. George
Schiller Park IL

I doubt if it'll be necessary, Wil, but just in case it is, don't ever hesitate to drop us a line and remind us that it's not a Ph.D.-level magazine, okay? Sorry about page 39. That's probably the only mistake we made in the first issue! Okay on the sweepstakes ... your name is entered. And thanks for the suggestion (in your reader's poll info) regarding an article on using a computer to aid in troubleshooting another computer. Fantastic idea. – John.

CONTROL APPLICATIONS

I have subscribed to and recently received the first issue of *Kilobaud*. This issue is about as I expected and very satisfactory. However, I feel there is another important use for the microprocessor besides computing systems and hope you do not overlook this field in future issues. Control applications open another complete field of endeavor.

I am a radio amateur. I have no computing systems and do not plan any. I would like to design and build microprocessor based systems for such things as repeater controls, and keyboard entry and scanning systems for synthesized radio equipment. With this in mind I feel there is a need for more basic materials than the virtues of Tiny BASIC, BASIC or assembly routines. What is needed is instructional material for the beginner on how to configure control systems, how to go from a flowchart to programs in machine language, suggestions for entering data from touchtone type keyboards, methods of BCD to binary and binary to BCD conversion, use of latches in outputting data to displays or other devices and how to program PROMs.

As you can tell from the above I am requesting articles on very basic subjects. I am sure there are many of your potential readers who have no experience in the art of software development such as myself. Further, my interest in control doesn't require I know all the high level routines. If such basic instructional material as has been suggested could in some way be incorporated into your publication it would seem a new enlarged group of subscribers would be open to *Kilobaud*.

Richard H. Pederson
Marshalltown IA

P.S. Possibly at a later date I will become interested in a small computing system.

Authors – take note! Microprocessors as we know them evolved from the need for sophisticated control systems. The F-8, for example, was developed for exactly the type of applications mentioned by Richard. As far as PROM programmers go, everyone seems to have a pet circuit, possibly interested readers will submit theirs. – John.

LATE DELIVERIES ON ISSUE #1

1 Jan. 1977

I'm writing you in concern over a subscription I purchased from you several months ago to *"Kilobyte."* Although I have seen no formal letter or note I assume that the magazine name has changed to *Kilobaud*. Is this true?

Now, if in fact I'm actually subscribing to *Kilobaud, where is it?* I was told when I paid for the subscription that the first issue would be out in mid-December. This was only partially true!

There is little else about a magazine that can make me as frustrated as having a subscription and seeing the latest issue at the newsstand days before I get mine in the mail. The Computer Center in San Diego has had their copies for sale for several weeks now and no longer have any copies left. I could understand *all* 1st issues being late but neglecting your charter subscribers is not a good way to kick things off. My check for twenty-five dollars was written in good faith – please don't abuse it.

I'm only one of many in San Diego still waiting.

Larry K. Bosworth
San Diego CA

Larry, I want to take this opportunity to apologize to everyone for the delay in the first issue. I thought for awhile it was due to the holiday mail but they continued to trickle in late out here on the West Coast for quite some time afterwards. If there was a snafu on our part, on the printer's part, or because of the U.S. Postal "Service" then we'll make every effort to correct it. It's a heck of a way to get started, and once again, we are sorry it happened. – John.

FEEDBACK

Issue #1 is outstanding in its content and focus! You have set a high standard for your readers to compare future issues against. While I am new to thinking "micro," I have worked professionally for fifteen years in the

continued on page 22

Rich Didday
1218 Broadway
Santa Cruz CA

IF COMPUTER SIMULATION IS THE ANSWER, WHAT'S THE QUESTION?

All aspects of the hobby/home computer movement's effect on our society are dependent on how widespread hobby/home computing becomes. If only a relatively small fraction of the households in the country ever get general purpose computers, then no matter how radical the effect is on those households, the overall effects will be small, and vice versa. How can we go about making educated guesses of the number of people who will have hobby/home computers in the years ahead?

There are several time-honored ways of approaching the problem of predicting the future behavior of extremely complex systems. The most popular is to ignore the problem as long as possible. (I'm reminded of what my old friend Walter Orvedahl used to say when faced by a maddening, intermittent hardware bug. "Don't worry, it'll get worse . . . and then you'll be able to figure out what's happening.") The second most popular way is to talk about the problem a lot, and bemoan the fact that nobody's doing anything about it. A third alternative is to wade in, accept that you're in for some rough weather, and try to find ways to deal with the problem. In our case, that means coming up with some preliminary models, accepting that they're going to be crude at first, and playing around trying to refine them until we get something that seems reasonable.

One technique that seems particularly appropriate to our situation is computer simulation. There are two main classes of computer simulation techniques, which go under the terms *structural models* and *aggregated models*. Structural models are best suited to situations in which you have a good understanding of the underlying processes. You proceed by choosing a representation for each subpart of the system under study, and by defining rules which describe how the subparts interrelate and interact. To make predictions from your model, you start it off in some configuration (initial state), and observe the changes that occur as the interaction rules are carried out. This is a particularly powerful and useful thing to do in some cases (for example, if you were considering building a super computer by hanging 137 Intel 8080's on the same bus, you could save yourself a lot of trouble and money by simulating your super computer, writing programs for the simulated monster machine, and discovering the good and bad points of your design before you ever touch a soldering iron). In other cases, however, it can be much too unwieldy. For example, if we were to use a structural model to get an idea of how many computer hobbyists there might be in a few years, we'd have to represent each person in the society, model their interactions with their neighbors and with the media, and, I suppose, model their thought processes to tell if they were going to buy a microcomputer. Obviously, that's out of the question, which leaves us with . . .

AGGREGATED MODELS

An aggregated model is cast in terms of *measurements* on the system under study instead of subparts of the system. In our case, instead of dealing with each person in the society, we can deal with the *proportion* of people in the society who have a home computer. Notice the difference. You can't touch a proportion like you can a person; a proportion is a measurement on (or a property of) the system we want to study.

An aggregated model is a little farther from reality than a structural model, but since there *is* a real underlying process (the interactions of the people in the society), we hypothesize (read "hope") that measurements on the whole society will be related in orderly ways, and that we can study those instead of worrying about each individual person.

Let's start by concentrating on just one property, namely the number of people who have a hobby computer. That's a quantity which has been growing fairly rapidly, and which we expect to keep growing. It happens that an incredibly wide range of real world systems (ranging all the way from yeast cultures in Petri dishes, to population of rabbits, to human populations) seem to grow in a similar way. Early phases of growth tend to be exponential. Later in the growth cycle, limiting factors tend to become more and more dominant and the growth rate tapers off, perhaps leaving the population at some stable, steady level. In the case of yeast cultures and human population, the availability of physical space and nutrient supply sets upper bounds on the total population. In the case of the number of computer hobbyists, the factors which determine the upper limit are less obvious. In fact, *none* of the parameters we need to apply the classical growth model to our problem is particularly easy to find. But the advantage of computer simulation is that we can easily and quickly try a wide range of parameter values in the hope of discovering reasonable values for them. Then we can play around a little more, making slight variations in our parameter values in an attempt to determine how sensitive our results are to errors in the parameter values. If variations in a particular parameter (say, the initial number of hobbyists) lead to large changes in our results, we can spend more effort trying to determine the value of that parameter.

A DISCLAIMER

Before we go plunging into the details of the model, let me emphasize that what we're doing here is preliminary, rough, and is not guaranteed to work, even though it does seem like the most promising alternative. One problem is my inability to find accurate statistics on the number of hobbyists already active. Needless to say, estimating the upper limit on the number of computer hobbyists is even more difficult. My hope is that presenting the model in a form that will run on virtually any system that has BASIC (see Program A) will encourage a large number of people to play around with it, including people who have access to inside information. If enough people try enough different parameter values and send in their results, we should, at least, be able to find *reasonable* parameter values.

The other problem is whether the classical growth model really applies to a population that is heavily media dependent. In the case of rabbits, for example, we can be fairly sure that new rabbits are the direct result of interactions among already existing rabbits. Thus, the idea that each year the number of rabbits increases in proportion to the number of rabbits alive at the beginning of the year is reasonable. But where do new computer hobbyists come from? What if people become hobbyists mainly because of things they read in magazines like *Kilobaud*? Then would it be more reasonable to try to relate the number of hobbyists to the number of computer oriented magazines printed each month? Please share your thoughts about this and other problems you detect.

THE LOOKING AHEAD GROWTH SIMULATOR

A BASIC program which simulates classical ("logistic") growth appears in Program A. Hopefully, the remarks make the program easy to understand. Most of the program is concerned with such matters as getting the parameter values and printing the results.

The key part of the program is the main loop – lines 3000 through 4090. And the key part of the main loop is line 3270. Each time period, line 3270, increases the number of hobbyists (H) by a factor of (HO/T)*(C-H)/C. The first term is the base growth rate (the number of new hobbyists per current hobbyist per year) divided by the number of periods a year. If only the first term were included, i.e., if the line read,

```
3270 LET H=H*(1 + (HO/T))
```

that would give pure exponential growth (try it). The (C-H)/C term slows the overall growth rate as H (the number of active hobbyists) approaches C (the maximum number of hobbyists). So overall, we get S-shaped curves – exponential growth at first, and then a gradual slowing down as the upper limit is approached.

Fig. 1 shows a run in which I assumed that there were 500 computer hobbyists in January, 1975; that base growth rate (why 3? I don't know); and the upper limit on the number of hobbyists was 500,000. I got the 500,000 figure by taking the circultation figures of the major computer hobbyists magazines, doubling it, and rounding up to an even figure (real scientific, right?).

In Fig. 2 I kept all the parameters the same as in Fig. 1 except for the maximum number of hobbyists. I set that at 50,000 just to see what differences it would make. By reading the numbers to the right of the plots, you can see that the number of hobbyists stays about the same in the two runs up to late 1975, and from then on, there's an increasingly great disparity.

In Fig. 3 all the parameters are the same as in Fig. 1 except for the base growth rate. Notice that cutting the growth rate by a third delays the final stage by only about one year.

EXPANDING THE BASIC MODEL

Since home computers will (presumably) get increasingly easy to use, since (some people believe that) home computers will come down in price, and since more and more people will be exposed to computers in high school, it seems likely that the upper limit on the number of potential owners of hobby/home computers will increase over time instead of staying constant as the basic model assumes.

To show how easy the basic program is to expand, I altered it to incorporate an exponentially growing maximum market size. I typed in the lines in Program B and ran the program twice to get the data which resulted in Figs. 4 and 5.

So, if you get tired of exploring the

```
100  REM :LOOKAHEAD SIMULATOR
110  REM :USED TO MODEL THE GROWTH
120  REM :OF HOBBY/HOME COMPUTING.
130  REM
140  REM :PARAMETERS--SET FOR YOUR
150  REM :SYSTEM AND DESIRES.
160  REM :"W"--NUMBER OF COLUMNS ON
170  REM :     YOUR OUTPUT DEVICE
180  REM :     (E.G. 72 ON MOST TELETYPES)
190  LET W=64
200  REM :"T"--NUMBER OF TIMES PER YEAR
210  REM :     TO PRINT STATUS (E.G. 12
220  REM :     FOR MONTHLY)
230  LET T=12
240  REM :"LO"--MAX # OF LINES OF OUTPUT
250  REM :     YOU'LL PUT UP WITH.
260  LET LO=50
270  REM :"L"--LINE COUNT.
280  LET L=0
300  REM :PARAMETERS USED IN MODEL.
310  REM :"Y"--INITIAL YEAR (E.G. 1973)
320  REM :"H"--NUMBER OF PEOPLE WHO HAVE
330  REM :     A HOBBY/HOME COMPUTER
340  REM :     (I.E. THE "POPULATION")
350  REM :"HO"--BASE GROWTH RATE (# OF
360  REM :      NEW HOBBYISTS PER CURRENT
370  REM :      HOBBYIST PER YEAR)
380  REM :"C"--CAPACITY--TOTAL # OF
390  REM :     PEOPLE WHO MIGHT SOMEDAY
400  REM :     GET A COMPUTER IF EXPOSED.
410  REM
1000 REM :GET INITIAL PARAMETER VALUES
1010 PRINT "WHAT YEAR DOES THE GROWTH START";
1020 INPUT Y
1100 PRINT "WHAT'S THE INITIAL # OF HOBBYISTS";
1110 INPUT H
1120 IF H>0 THEN 1200
1130 PRINT "POSITIVE VALUES ONLY, PLEASE."
1140 GO TO 1100
1200 PRINT "WHAT'S THE POTENTIAL # OF HOBBYISTS";
1210 INPUT C
1220 IF C>=H THEN 1250
1230 PRINT "THERE ARE ALREADY MORE THAN THAT"
1240 GO TO 1200
1250 PRINT "AND WHAT'S THE GROWTH FACTOR THE"
1260 PRINT "FIRST YEAR";
1270 INPUT HO
1280 REM
2000 REM :PRINT HEADINGS FOR PLOT
2010 PRINT
2020 PRINT "GROWTH CURVE STARTING WITH ";H
2030 PRINT "HOBBYISTS IN ";Y;" WITH A BASE"
2040 PRINT "GROWTH FACTOR OF ";HO
2050 PRINT "MAX. # OF HOBBYISTS=";C
2100 PRINT
2110 PRINT " YEAR MIN";
2120 FOR B=1 TO W-23
2130   PRINT " ";
2140   NEXT B
2150 PRINT "MAX VALUE"
2160 REM
3000 REM :MAIN LOOP--DO ONCE EACH PERIOD
3010 REM :(I.E. "T" TIMES PER YEAR), UNTIL
3020 REM : POPULATION IS WITHIN 1% OF "C"
3030 REM : OR LINE LIMIT IS EXCEEDED.
3040 REM :COUNT PERIODS
3050 LET P=1
3060 REM :PRINT CURRENT SITUATION.
3070 GOSUB 10000
3250 REM
3260 REM :UPDATE FOR NEXT PERIOD.
3270 LET H=H*(1 + (HO/T)*(C-H)/C)
3310 REM :LOTS MORE ROOM HERE FOR
3320 REM :MORE EQUATIONS
4000 LET P=P + 1
4010 IF P<=T THEN 4050
4020   REM :DONE WITH THIS YEAR.
4030   LET Y=Y + 1
4040   LET P=1
4050 REM :BOTTOM OF MAIN LOOP.
4060 REM :QUIT IF WITHIN 1% OF MAX
4070 REM :OR LINE LIMIT EXCEEDED.
4080 IF L>LO THEN 5000
4090 IF H<.99*C THEN 3060
5000 REM :PRINT THE LAST LINE
5010 GOSUB 10000
6000 PRINT
6010 PRINT "WANT TO TRY AGAIN? (YES OR NO)";
6020 REM :IF YOUR VERSION OF BASIC DOESN'T
6030 REM :ALLOW YOU TO INPUT STRING VALUES,
6040 REM :CHANGE TO (1 FOR YES, 0 FOR NO) FORM.
6050 INPUT A$
6060 IF A$="YES" THEN 100
6070 IF A$="Y" THEN 100
6080 GO TO 20000
10000 REM :PRINT SUBROUTINE
10010 IF P>1 THEN 10100
10020   PRINT Y;":";
10030   GO TO 10110
10100 PRINT "      :";
10110 REM :SEE HOW MANY *'S TO PRINT.
10120 LET S=INT((W-17)*H/C + 1)
10130 IF S<=W-17 THEN 10200
10140 LET S=W-17
10200 FOR B=2
10200 FOR B=1 TO S
10210   PRINT "*";
10220   NEXT B
10230 FOR B=S + 1 TO W-17
10240   PRINT " ";
10250   NEXT B
10260 PRINT INT(H)
10270 LET L=L + 1
10280 RETURN
20000 END
```

Program A

```
GROWTH CURVE STARTING WITH  500
HOBBYISTS IN  1975  WITH A BASE
GROWTH FACTOR OF  3
MAX. # OF HOBBYISTS= 500000

YEAR  MIN                                            MAX VALUE
1975  :*                                               500
      :*                                               624
      :*                                               780
      :*                                               975
      :*                                               1219
      :*                                               1523
      :*                                               1903
      :*                                               2377
      :*                                               2968
      :*                                               3706
      :*                                               4625
      :*                                               5771
1976  :*                                               7197
      :*                                               8971
      :**                                              11173
      :**                                              13905
      :**                                              17284
      :***                                             21456
      :***                                             26590
      :****                                            32884
      :****                                            40564
      :*****                                           49883
      :******                                          61109
      :********                                        74520
1977  :*********                                       90373
      :***********                                     108883
      :*************                                   130176
      :***************                                 154247
      :******************                              180912
      :********************                            209776
      :***********************                         240217
      :**************************                      271419
      :*****************************                   302440
      :********************************                332315
      :**********************************              360177
      :*************************************           385357
1978  :***************************************         407446
      :*****************************************       426302
      :******************************************      442010
      :*******************************************     454826
      :********************************************    465099
      :*********************************************   473215
      :**********************************************  479553
      :**********************************************  484455
      :**********************************************  488221
      :*********************************************** 491096
      :*********************************************** 493282
      :*********************************************** 494939
1979  :*********************************************** 496191
```

Fig. 1.

```
GROWTH CURVE STARTING WITH  500
HOBBYISTS IN  1975  WITH A BASE
GROWTH FACTOR OF  3
MAX. # OF HOBBYISTS= 50000

YEAR  MIN                                            MAX VALUE
1975  :*                                               500
      :*                                               623
      :*                                               777
      :*                                               969
      :**                                              1206
      :**                                              1501
      :**                                              1865
      :***                                             2314
      :***                                             2865
      :****                                            3541
      :*****                                           4363
      :******                                          5359
1976  :*******                                         6555
      :********                                        7979
      :**********                                      9656
      :***********                                     11604
      :**************                                  13832
      :****************                                16333
      :******************                              19082
      :*********************                           22032
      :************************                        25113
      :***************************                     28238
      :******************************                  31311
      :*********************************               34237
1977  :***********************************             36935
      :*************************************           39348
      :***************************************         41443
      :*****************************************       43216
      :*******************************************     44682
      :********************************************    45870
      :*********************************************   46817
      :*********************************************   47562
      :**********************************************  48142
      :**********************************************  48589
      :**********************************************  48932
      :*********************************************** 49193
1978  :*********************************************** 49391
      :*********************************************** 49541
```

Fig. 2.

```
GROWTH CURVE STARTING WITH 500
HOBBYISTS IN 1975 WITH A BASE
GROWTH FACTOR OF 2
MAX. # OF HOBBYISTS = 500000

YEAR   MIN                                              MAX VALUE
1975   :* 500
       :* 583
       :* 680
       :* 793
       :* 925
       :* 1079
       :* 1259
       :* 1468
       :* 1712
       :* 1996
       :* 2328
       :* 2714
1976   :* 3164
       :* 3688
       :* 4299
       :* 5009
       :* 5836
       :* 6797
       :* 7914
       :* 9213
       :** 10720
       :** 12468
       :** 14495
       :** 16840
1977   :** 19553
       :*** 22684
       :*** 26293
       :*** 30445
       :**** 35211
       :**** 40666
       :***** 46892
       :****** 53975
       :****** 62000
       :******* 71052
       :******** 81211
       :********* 92548
1978   :********** 105117
       :************ 118954
       :************* 134063
       :*************** 150415
       :**************** 167943
       :****************** 186532
       :******************** 206023
       :********************** 226211
       :************************ 246856
       :************************** 267686
       :**************************** 288415
       :****************************** 308757
1979   :******************************* 328439
       :********************************* 347221
       :*********************************** 364904
       :************************************ 381336
       :************************************** 396420
       :*************************************** 410107
       :**************************************** 422395
       :***************************************** 433322
       :****************************************** 442953
       :******************************************* 451376
       :******************************************** 458692
       :******************************************** 465007
1980   :********************************************* 470431
       :********************************************* 475068
       :********************************************** 479016
       :********************************************** 482366
       :********************************************** 485202
       :********************************************** 487595
       :*********************************************** 489611
       :*********************************************** 491307
       :*********************************************** 492730
       :*********************************************** 493924
       :*********************************************** 494924
       :*********************************************** 495762

WANT TO TRY AGAIN? (YES OR NO)? NO THANKS
```

Fig. 3.

```
DELETE 2110-2150
DELETE 10110-10260
380  REM :"C"--INITIAL CAPACITY (MAX. NUMBER OF HOBBYISTS)
390  REM :"CO"--GROWTH FACTOR FOR MAX. NUMBER OF
400  REM :        HOBBYISTS
1280 PRINT "WHAT'S THE GROWTH RATE OF THE MARKET";
1290 INPUT CO

2060 PRINT "AND THE MAX. NUMBER OF HOBBYISTS GROWS"
2070 PRINT "BY A FACTOR OF ";CO;" EACH YEAR."

3280 LET C=C*(1 + CO/T)

10110 PRINT H," ",C
```

Program B

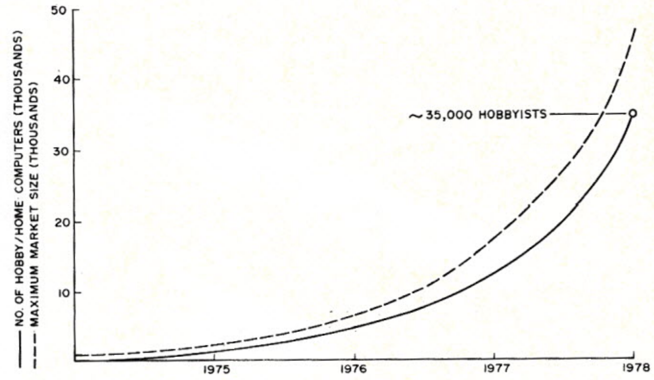


Fig. 4. If the growth rate of the number of hobbyists is large compared to the growth of the maximum market size, the number of hobbyists just tracks the maximum market size. Second version of the simulator; initial number of hobbyists = 100, growth factor = 4, initial market size = 100, growth factor = 1.

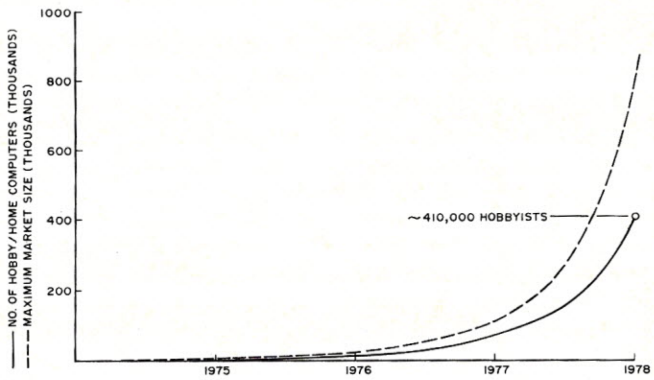
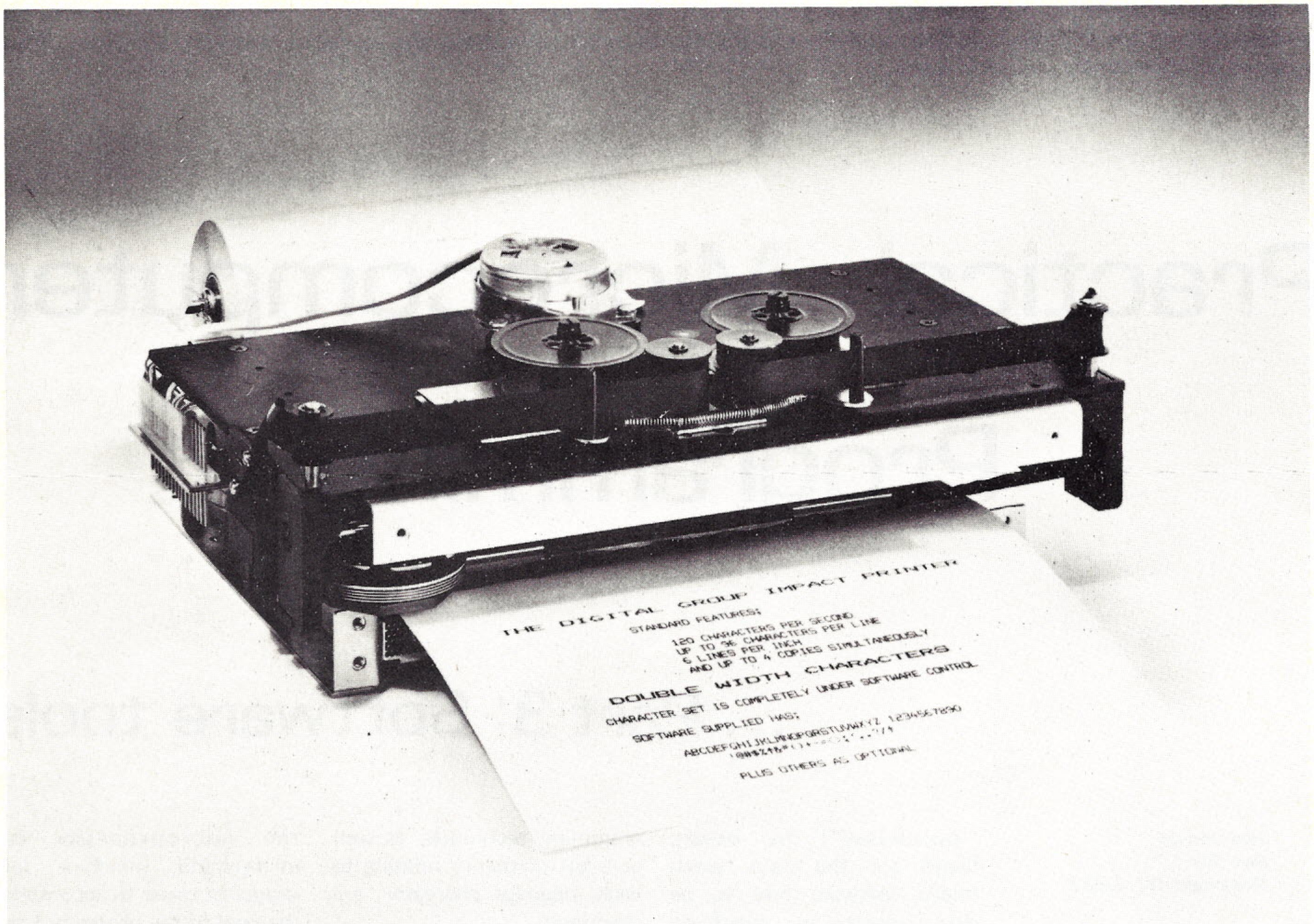


Fig. 5. Another run of the second version. Parameters are the same as in Fig. 4 except market size growth rate is doubled and initial market size is halved. Results are roughly the same until mid-1976, by 1978 these parameters give more than 10 times as many hobbyists/home computers.

behavior of the basic model (Program A), devise your own improvements. If you find them useful and/or interesting, please send them in for us to share.

Write:

Looking Ahead
1218 Broadway
Santa Cruz CA 95062.



Print Your Heart Out.

With help from the Digital Group, naturally.

Now, that small computer system you own or have been considering for personal or business use suddenly becomes a lot more usable—with the addition of a full-size *impact* printer from the Digital Group. A printer designed for small computers that need big output (like yours).

With the Digital Group printer, you can print your heart out...and it won't cost an arm and a leg. The Digital Group printer is available for less than $500. That's right—$500.

Just look at these specifications:

- Fast—120 characters per second
- 96 characters per line
- 12 characters per inch horizontal
- 6 lines per inch
- Makes up to 4 copies simultaneously
- Character set and pitch variable under software control—double width characters, etc.
- 5 x 7 character matrix
- Ribbon has built-in re-inkers for a life of 10,000,000 characters
- Paper can be either a standard 8½-inch roll, fanfold or cut page
- Interfaces to 8-bit parallel ports

There are lots of capabilities and outstanding features of the Digital Group printer...and (as always) the best news is our price. Kit prices start as low as $495 for the printer and interface card. It simply can't be beat.

Find out all the facts about the Digital Group printer now. Just fill in the coupon below or give us a call for the details. We think you'll find a place for our printer in your system ...and in your heart.

the digital group

P.O. Box 6528
Denver, Colorado 80206
(303) 777-7133

Quick. I want to print my heart out.
Send me all the details on your full-size impact printer.

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

Please print.

As usual, John is putting out some quality material which, in this case, is aimed primarily at you hardware types who have wondered about the difference between assemblers and cross-assemblers, simulators and emulators, and etc. (The "etc." should take care of anything we missed.) – John.

Practical Microcomputer Programming

... Part 3: Software tools

John Molnar
Box 561
Ridgefield NJ 07657

The software revolution is upon us! Personal computing is entering a new phase based upon the increasing availability of software (programs) for microcomputer systems. As recently as a few months ago the only *commercially* available software for micros consisted of assemblers, collections of utility routines, and a BASIC language interpreter or two.

Check out today's advertisements, however, and note how often *software* is mentioned in the same line as the hardware features of a given micro. Now several major vendors offer different versions of BASIC, most have assemblers, and the other day I even noticed an ad for a complete FORTRAN compiler for the 8080 in a computer trade paper!

What's more, the manufacturers are not the only ones producing micro software. Every home system is a potential source of programs that can be used (read "purchased") by others. Gone are the days when micro software had to be developed in machine language or assembled on small, vendor supplied, resident assembler programs.

A whole range of software *tools* will become available to the home computer enthusiast in the months and years to come. Some of us are familiar with BASIC interpreters, but what are these *compilers?* And what about *cross-assemblers?* Ever heard of an *emulator?* There are many ways of developing micro software that do not require the tedious *hand-assembly* techniques familiar to all micro programming pioneers.

This article explores several software development aids that are not presently commonplace in the micro world. However, time flies, and soon these *new* products will be appearing in tomorrow's micro system advertisements. (Who would have predicted FORTRAN for the 8080 a year or so ago?) Several software *tools* will be discussed, and the benefits and trade-offs of each programming technique, as well as a bit of theory relating to each *language processor*, are described.

The purpose of this article is not to describe specific languages, but rather to introduce the reader to methods of developing software that are sure to become more widely known in the future.

Why Languages?

Anyone who has ever hand-assembled a micro program and then done it over to correct a minor bug knows the advantages offered by a *language processor* such as BASIC or even assemblers. The BASIC language allows English-like programming, as well as the capability to easily correct a program bug.

There are, however, other methods that allow software to be easily generated for micro systems. One of these is the *compiler,* of which the FORTRAN language is an example. There are also *cross-assemblers* and *emulators* which allow micro software to be generated on other computer systems.

The end result of any language is to produce code that the microprocessor can understand (machine language) in order to accomplish the goal of the program. Let's take a look at several language processors and see how they produce the code that can run on *your* system.

The Assembler – A Step Above Machine Language Programming

The language most familiar to micro programmers is the *assembler*, a program that converts the English equivalent of machine instructions into actual machine code on a one-for-one basis. For example, the expression "CLR A" in a Motorola 6800 assembly program is converted to the hexadecimal code "4F", which clears the "A" register when executed. The assembler also allows symbolic references to memory locations, so the programmer does not have to concern himself with actual memory locations while writing a program.

So instead of saying "CLR #1000" the programmer can write "CLR BUFF", where "BUFF" is really the symbolic equivalent of location

```
LOCATION   CODE       INSTRUCTION     COMMENT

0000       4F         CLR A           Clear A register
0001       5F         CLR B           And B register.
0002       7F 10 00   CLR $1000       Also clear loc. 1000..
0005       7F 20 00   CLR BUFF        And first buffer location.
0008       3F         SWI             Halt
                      .
                      .
                      .
               BUFF EQU $2000         Equate symbol to actual value
```

Program A. A simple assembly language program for the M6800. Note that each line of user source code generates a single machine instruction. The assembler allows symbolic references to actual memory locations, which eliminates the need to know specific memory addresses while writing an application.

1000. The assembler also allows comments to be written into a listing so other programmers can follow the logic flow of an application. (Always a good idea!) Program A illustrates a simple assembly program that clears the three registers and location 1000 in a M6800 system.

At this point a few terms must be defined that apply to language processing programs. The input to a language processor, be it an assembler, BASIC, or whatever, is called the *source program.* The source program may be a deck of cards, a paper tape, or a file on a floppy disk. A source program is, of course, generated by a programmer. The output of an assembly process is known as *object code*, that is, code which executes on a specific system.

Most assemblers available for personal systems produce code to be run on the same type of target computer. The assembler itself is written in the machine language of the micro it supports. Thus, the micro must have enough memory and peripherals available to support the assembly process. Usually 8K of random access memory (RAM) is required, as well as a console device (CRT or TTY), and some type of output device for the object code generated by the assembler. In many instances, however, the memory and peripherals required to perform the assembly are not present on the micro system. This problem caused the development of another type of assembler, the *cross-assembler.*

The Cross-Assembler – Produces Code for a Micro While Executing on a Larger System

Needless to say, assemblers are long (about 6-8K) and logically complex. Consider the problem of writing the original assembler for any micro. This assembler could have been (and often was) coded by hand in machine language with all the associated problems of debugging and modification, or it could have been written using another large computer possessing an assembler and high speed peripherals. In the latter case the object code produced does not run on the *host* computer, but rather on the micro system for which the code is targeted. This process is known as *cross-assembly*, and the technique is used to produce most of the *vendor supplied* micro software used by home systems. Remember that the code produced by a cross-assembler does not run on the machine that produced it but rather on a different system.

The advantages to this approach are obvious. The cross-assembler usually has extensive error handling routines, good diagnostic messages, and offers an array of features to the programmer not found on micro assemblers. And of course the target micro does not care where the code it runs is produced. Cross-assemblers have always been around, as several years ago large mainframe computers ran cross-assemblies for the minis that now support our micro based systems.

Most of the large micro chip manufacturers have cross-assemblers available for a number of mini and mainframe systems. Intel and Motorola have cross-assemblers on several timesharing services, and I have seen several privately produced cross-assemblers that run on minicomputers.

There is no reason why universities and large machine users cannot obtain and support cross-assemblers for micros. It is entirely possible for computer groups and clubs to get together with the local community college or university in order to cross-assemble large programs that could not be assembled using a small, micro based, resident assembler. In fact, a cross-assembler for the M6800, for example, could be developed to run on a *loaded* Altair 8080 system. (Bored Altair users – start thinking!)

It seems that many micro users are not interested in using assembly language for developing large micro systems. The introduction of BASIC to the micro world opened the door to a whole new software ball game – that of using *high-level* languages for English-like programming.

However, some confusion exists concerning the advantages and trade-offs of BASIC programming, and the eventual use of FORTRAN and other compiled languages is sure to compound the misunderstanding about these languages. The terms *compiler* and *interpreter* are often confused in discussions about high-level languages.

Hopefully the following discussion will clarify the differences between the two approaches used to generate English-like languages. The interpreter is discussed first, as most programmers are already familiar with BASIC, the best known interpreter.

The Interpreter – Allows English-Like Programming

The interpreter is a program that converts English expressions into machine responses. Program B illustrates a BASIC program to solve the Ohm's Law relationship E=I*R. Note that the source program is written in algebraic form understandable by nonprogrammers. Program C is an assembly language program written in M6800 code to solve the same relationship. Recalling that the assembler converts source statements into machine code on a one-for-one basis, one can see that the BASIC program requires some sort of intermediate processing to convert it into machine code. (The micro that can understand the expression E=I*R is a long way off!)

The *interpreter* is the vehicle that converts the user source into a form understandable by the micro. This is accomplished by scanning and analyzing (interpreting) each user source line and directing specific internal routines in the interpreter program to produce the desired results. For example, to process the single expression E=I*R, the BASIC interpreter must first look up the values of I and R, multiply them, and place the result in a memory location known as "E". Obviously, this takes several machine instructions, exactly as did the assembly language solution. The main difference is that in one case the programmer had to define each machine step (assembly) and in the other the interpreter program caused the correct machine action based on the user's source statement.

The single most significant feature of the interpreter program is that it *does not* produce any object code to solve the problem, as that code is *built into* the interpreter and is only summoned by user source instructions.

Hopefully it can be seen that the assembly language output of Program C is not produced by the interpreter, even though the end result of the two programs is the same. In the case of the interpreter, the correct result is achieved by summoning internal routines within the interpreter. The user program is never *in control* of the micro, as the interpreter code is actually executing at all times. Thus the interpreter must always be present in memory while BASIC is executing.

Contrast this action to that of the assembler. After an assembly, the object module produced is executed on the machine, not the assembler. In fact, the assembler does not have to be in memory once its task is complete. However, in order to run the same assembly application on another family of microprocessors the assembly source must be recoded in the new assembly language and reassembled to produce a new object module.

Not so with BASIC source, however! Recalling that the source for an interpreter is not directly executed by the micro, but only directs interpreter action, it can be seen that BASIC programs are transportable between different micro families, a super advantage. If specific conventions are followed in writing the source, a given BASIC program could run on a 6800 system as well as on a 8080 based micro, which is a real boon to those writing software for distribution or sale.

There are, however, several disadvantages in using an interpreter such as BASIC. The most significant shortcoming is in the area of memory. The interpreter must have within itself all the routines necessary to execute every allowable user source statement, even though many of them are never called. For example, a program written to solve Ohm's Law does not require the math functions SIN or COS allowed by BASIC, but a trigonometry problem does. Of course, the routine in BASIC to interpret the math functions must be present at all times, as the interpreter must be capable of handling any user program. The memory used by routines that are not called is often considered to be *wasted*, although nothing can be done about it.

Contrast the interpreter with the assembler, which is not required in memory after the user source is processed. The output object code from the assembler contains exactly what the user specified, nothing more or less. (Bugs not considered!) The BASIC interpreters now available occupy about 4-8K, and all user programs and the space required to hold data, etc., adds to the size of the interpreter. However, the memory overhead of the interpreter is rapidly forgotten by the user when the advantages of programming in an English-like language are considered.

The second shortcoming of the interpreter is that a user source program will *run*, or be interpreted at a rate 10 to 50 times slower than an assembly language equivalent. This time overhead is the result of the steps the interpreter must take while evaluating, decoding, and executing user source statements. If any of the restrictions of the interpreter are intolerable, the user may desire to use another English-like language processor, the *compiler.*

STATEMENT	EXPRESSION
0001	REM A SIMPLE BASIC PROGRAM TO CALCULATE THE
0005	REM OHMS LAW RELATIONSHIP. INITIAL VALUES FOLLOW:
0010	LET I=5
0015	LET R=22
0020	LET E=I*R
0030	PRINT "THE ANSWER IS:",E
0040	END

Program B. A BASIC language to solve Ohm's Law. Note that source statements are written in English-like expressions. The interpreter takes care of assigning memory locations and program control. Contrast this approach to the assembly language example of Program C which does exactly the same thing.

LOCATION	CODE	INSTRUCTION	COMMENT
0000	86 05	LDAA #5	The value "5" to A register
0002	C6 16	LDAB #22	The value "22" to the B register
0004	BD 10 00	JSR MULT	Go multiply to produce "E"
0007	BD 20 00	JSR PRIN	Go print result
000A	3F	SWI	Halt
		.	
		.	
1000		MULT	(A user written multiply routine as the M6800 has no multiply capability)
2000		PRIN	(A user written output routine)
			END

Program C. An assembly program to solve Ohm's Law. Note that each step has to be defined in machine language that is not easily identified with the problem at hand. The user must also provide the special multiplication and output routines, as these are not provided on most micros in hardware. The above program is a simplification of the problem. A practical routine would have allowed the user to specify input data instead of coding the values into the program!

```
      EXPRESSION
C     A FORTRAN PROGRAM THAT SOLVES A RIGHT TRIANGLE PROBLEM
C     THE TRIG FUNCTION SIN IS CALLED BY THIS PROGRAM
      A = 56.12
      B = 3.14
      C = SIN(A) + SIN(B) -1.0
      TYPE A,B,C
      STOP
      END
```

Program D. A FORTRAN program that calls "external" functions. In this program output is produced consisting of machine code and references to the routines SIN and TYPE. The code for these external routines is contained in a RUN-TIME LIBRARY supplied with the compiler.

**The Compiler
– Generates Object Code
Directly From User Source**

The compiler is a language processor that evaluates algebraic source statements in to similar fashion to BASIC and other interpreters. However, the compiler *produces* object code to solve the user's problem. The output of the compiler is machine language similar to the output of the assembler program, contrasted to the interpreter which produces no output code. Often many lines of machine code are produced by processing a single user source statement.

Again, referring to Program B and Program C, note that the compiler must produce code similar to that

produced by the assembly language solution of the expression E=I*R. Another difference between the compiler and interpreter is that the former produces only the code needed to solve the user's problem.

In Program D, the FORTRAN program requires the SIN math function to solve a right triangle problem. The output code generated by FORTRAN does not, however, contain specific instructions to calculate the SIN. Instead, a *reference* to the required routine is output. This reference is usually referred to as an EXTRN (EXTeRNal reference). This reference must be *resolved* before the compiler output can be run on the target computer, thus introducing an extra step not required in assembly language or BASIC programming.

The process of executing compiler output works as follows. The compiler generates a *bare-bones* object module containing EXTRN references to any special routine required by the user. The computer has no way of executing the reference code, so an intermediate utility program called a LINK LOADER must be run, using the output of the compiler as its input. The function of the link loader is to scan the compiler output for any references to external routines. When one is found, the loader *links* the required program from a library of routines supplied with the compiler. This library of routines is often called a RUN-TIME library, as it consists of routines necessary to *run* the final program.

The final output of the link loader contains the original FORTRAN machine code plus the run-time routines linked in from the run-time library. This output is referred to as a LOAD MODULE, that may be executed in exactly the same fashion as assembler output. Fig. 1 graphically illustrates the compilation process, consisting of compiler execution, link-loader operation, and finally the execution of the user's program.

By now you are probably thinking that the operation of a compiler is highly complex. Compared to the interpreter it is, but only because an additional step is required to finally *run* the user program. Compiler programs and the associated run-time library are highly sophisticated software products requiring many man-months to produce, which is why not many have arrived on the micro scene. There are *mini* compilers which do not require a run-time library and link-loader, but these have not appeared in the micro picture either.

Let's look at the advantages of using a compiler if one is available. Of course, English-like statements are permitted as input, exactly as with an interpreter. However, the compiler usually allows many features that are not supported by the interpreter, as the compiler must only produce EXTRN references to features that would have to be contained within the interpreter, making it prohibitively large. The compiler is not required in memory after the user source has been processed, thus all the micro's memory is available to execute the object program.

The code generated by a compiler executes much faster than does the interpreter, recalling that compiler output is actually machine code comparable to that of the assembler.

And finally, the compiler, like the interpreter, allows program transportability between micros, as the compiler produces the code needed for the specific target microprocessor. The source program need never know what machine it is finally going to execute upon.

The Simulator — allows large machines to think like a micro

There is a final language

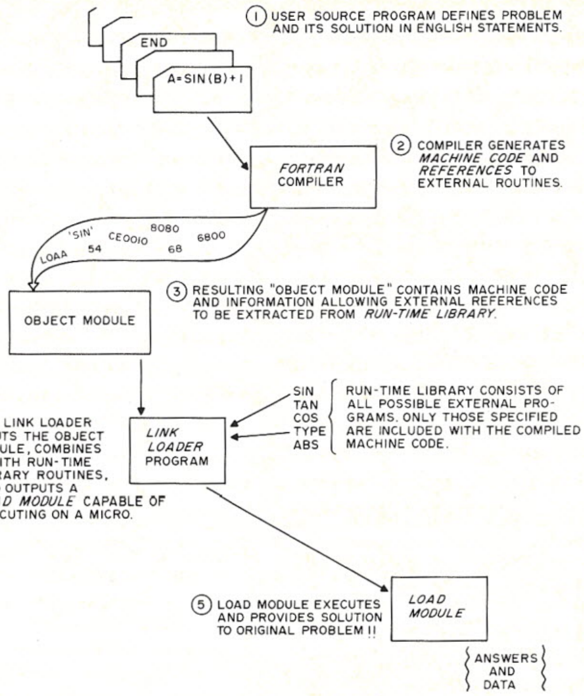


Fig. 1. The compiler process. In contrast to the assembler and interpreter the compiler must output references to external routines called by the source program. These references, such as the math function SIN and the output routine TYPE are contained in a library of such routines, the RUN-TIME LIBRARY. An additional utility program, the LINK-LOADER, must be run following the compiler step. The Link-Loader combines the compiler machine language output with the necessary library programs. The final LOAD MODULE, the output of the Link-Loader, may then be executed on the microprocessor system to provide the solution for the original problem.

processor that must be discussed in any general investigation of computer languages. Recall that the compiler and interpreter allow machine independence when writing programs, as long as the source statements follow a predefined set of rules. Of course, to run the program one must have a micro system capable of executing the object code produced by the language processor. Often, however, the required micro system may lack the memory or peripherals to *run* the program. Manufacturers of equipment employing micros are often faced with this problem — how does one determine which micro architecture is best for a given application without buying five or six evaluation systems? One of the solutions lies in the writing of a *simulator.*

The simulator program runs on a mini or mainframe system and allows a micro object program to *run* by simulating the instruction set of the micro. For example, a simulator for the National PACE micro executes by simulating each micro instruction with routines written in the machine language of the *host* computer. Thus, the simulator allows programmers to write and debug microcomputer programs without even using the system for which they are intended. Of course, simulating a micro instruction set is a slow process, and the simulated program usually executes more slowly than it would on an actual micro system.

The simulator program is really a type of interpreter, but, instead of processing English-like source statements, the input is the machine code (or assembly language source) of the micro being simulated. The simulator is probably not a tool within reach of the home micro experimenter, as a mini

or larger system is needed to make the simulator program reasonably efficient. Again, however, there is a possibility of clubs obtaining time on university or time-sharing systems capable of running simulators.

Another form of "simulation" is a program called an *emulator.* An emulator is a microprogram – called "firmware" – which is developed on a microprogrammable machine to emulate another machine. Certain minicomputers such as the Interdata 8/32, Varian V-70 Series, Hewlett-Packard 2100 and others are microprogrammable machines which have Writable Control Store. This Writable Control Store can be programmed – with a microprogram – which will enable this type of computer to directly execute machine code from another computer (e.g., the 8080). The subject is somewhat *heavy* and should be discussed in a separate article at another time. Suffice to say that a simulator will simulate the execution of programs from another computer (through software) and an emulator – running on a microprogrammable machine – will directly execute programs from another computer (through firmware).

Conclusion

The past two years have seen a tremendous outburst of micro software. Although most home systems rely upon assembly language and BASIC programming for software development, the day is not far off when compilers and *complex* mainframe and mini software will be available to the home enthusiast. To date several FORTRAN compilers have appeared, and it should not be long before other compiled languages and their utilities are commonplace.

Hopefully this discussion of computer languages and techniques will be helpful in understanding the terminology that is sure to appear in tomorrow's microprocessor advertisements. Remember that tomorrow is never far away in this field! ■

Letters to the Editor

from page 11

"maxi", "midi" and "mini" software areas. The trends are very favorable for the explosion of small business and home hobby computer use. Since you folks know this better than I, let me turn my attention to the purpose of this letter.

Feedback: I have spent several hours reading this first issue from cover to cover. What surprised me most was the total time I have spent reading this issue given the number of other publications I have on my desk and some small amount of work. So if feedback is what you want, let me start with the front cover. Don't change it! Save the artwork for somewhere else. I want to be able to find an article quickly when I am looking and your cover will speed my searches in the future.

The four column format is tight but I like it. This is especially true with technical articles which refer to figures and tables because I hate to flip pages to refer to something. The article mix and quality was very good. Since I have a very weak hardware background, the "Is the Z80 the Wave of the Present?" article and "Nobody Knows the Troubles I've Seen" piece were basic to my education process. The weakest article I feel is the "Computer Control of the World!" piece by Bowick. Too much hype and not enough practical content. (It also is the only double line in your cover index which might make the first article reviewed at a newsstand.) So please do not let the growth concept get out of control and permit yourselves to sensationalize this subject. The subject is sensational enough without pedalling it.

Software articles were fantastic. Pin a blue ribbon on Wilcox for his presentation in "The Hobbyist's Operating System" article. I very much look forward to Part 2 next month. "Software Exchange" by Childs brought to light the very real problems associated with technology transfer when different vendors, programmers, etc., are involved. I don't want to leave out Stark's "Programming? It's Simple!" but I cannot be objective here having been into it for so long.

Do I think the first issue was perfect? Not yet but it was close. Let me provide some practical which may be of assistance to future successes of *Kilobaud*. The sweepstakes is a great idea (I certainly plan to enter; maybe several times since this is not prohibited in the rules – there were no rules that I could find – just a March 15, 1977 deadline). I have faith in *Kilobaud's* management of the sweepstakes but take care to *document* how winners were chosen, procedures, no employee of *Kilobaud* could participate, etc. This will keep everyone happy.

What about a quarterly club directory *insert* to *Kilobaud*. This will help local clubs grow and find each other and increase local membership greatly. We had two clubs in Charlotte which found each other by accident and have since merged. Also a list of future events by clubs and/or national shows such as the NCC Dallas get together in June (13-16). I try to plan my travel to accomplish multiple goals and such a calendar would be helpful.

I would like to see articles like, "This is a Computer Store" which defines what goes into such a creation from a business perspective; "Servicing Your Computer and Peripherals" which addresses this very real problem for small business system users who are attracted to microsystems; "Business Applications – Parts 1, 2, 3, 4, 5, 6, . . ." which defines the basic set of user applications; and a "Trends Continued" column or article series which guestimates where we are going in the next five and ten years.

Time is the most limited resource of the microsystem user when the hobby is not the vocation. *Kilobaud* should focus on cutting time requirements in software development, hardware assembly, etc. This has been noted in the first issue but perhaps a point can be made of this goal by *Kilobaud* authors.

Mike Weaver
Charlotte NC

We welcome the good and the bad but naturally we really get a stroke of pleasure out of letters like yours, Mike. The suggestions are well taken . . . keep them coming. – John.

---

1 Print "Publisher's Remarks" 10 No End Run

from page 3

when the rig was modified and working okay . . . it was a problem on the first few sets. A later good review would never catch up with the bad one and the manufacturer would have lost the sale of several thousand sets . . . possibly a $1 million loss.

When I find problems such as that I get on the phone and talk them over with the manufacturer. It is seldom that he doesn't get things straightened out. Since I've a good rapport with the readers of *73*, I don't usually have to find out about product problems myself . . . normally I get a call or a letter from a reader about it. Again, I talk it over with the manufacturer. If I find him to be unreasonable I cut off advertising until things are straightened out. In the ham field there are over a dozen firms which are not permitted to advertise in 73. This costs us perhaps $5000 a month in lost ad revenue, for the firms are advertising in the other ham magazines.

In the computer field it is going to be a while before we are able to check out everything coming down the line . . . there's just too much of it and it is too expensive at about $2000 per system . . . and the time . . . ! We're doing what we can in our lab and beyond that we will have to depend upon the readers of *Kilobaud* to let us know about problems. We have the Altair, Imsai, Wave Mate, Sphere, Ebka, and Monolithic Systems here so far . . . with promises of a lot more to come.

My feeling is that it would take such a lab full of people to honestly review computer systems that it isn't practical right now. I think that the fellows who have developed the hardware are probably more experts on it than anyone else, so I've been trying to get these busy chaps to sit down and tell us why they designed their circuits the way they did . . . the benefits to all of us of this design . . . and the trade-offs involved. I fully expect some controversy to develop and I think we'll all be the winners as a result.

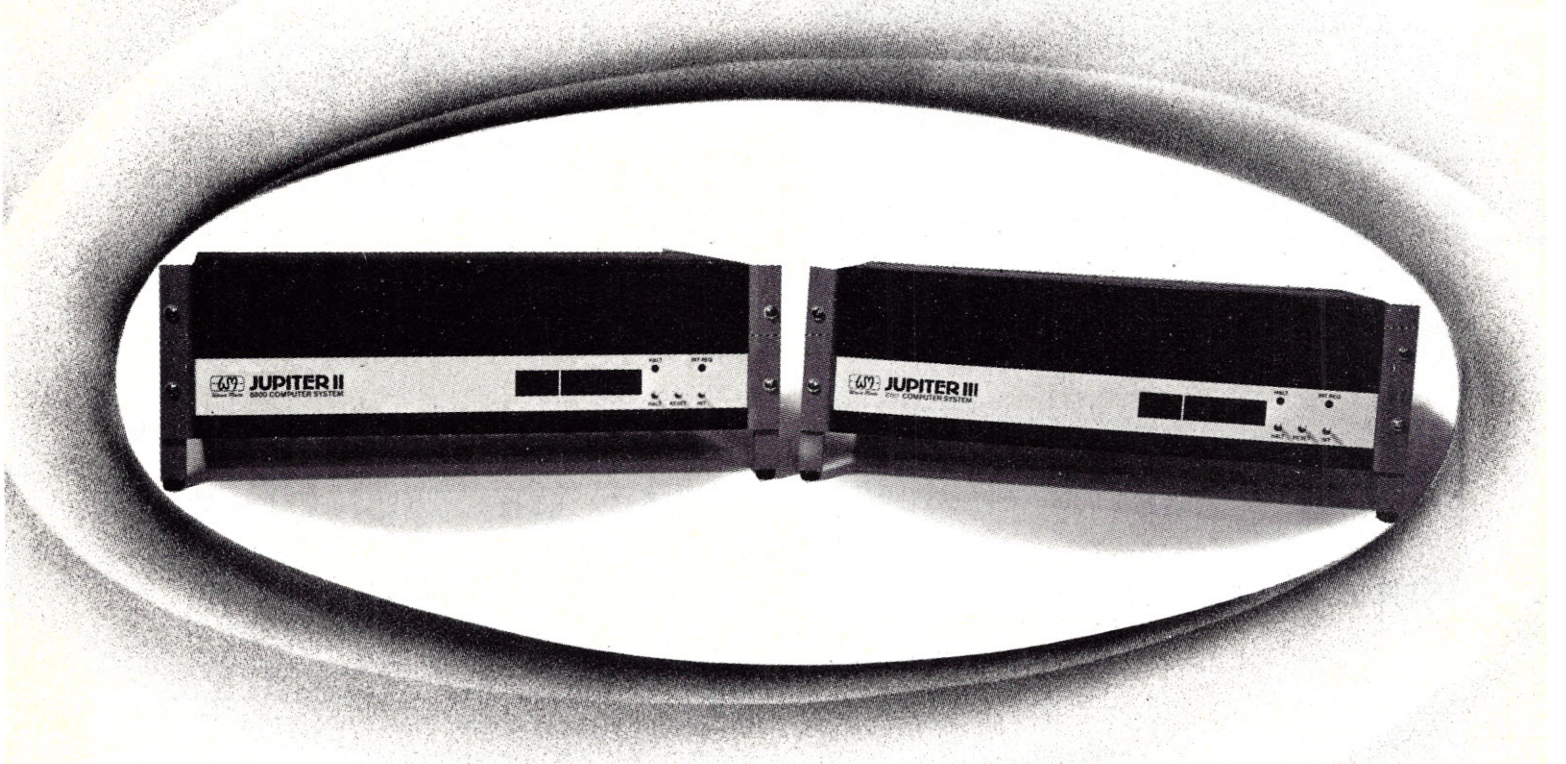
An example is the article in the second issue of *Kilobaud* by Hal Walker. Hal is certainly an expert in cassette systems, with his National Multiplex equipment. I'll be surprised if we don't get some good articles on the other cassette systems as a result of the criticisms. Maybe all this will help us toward some standardization such as I tried to achieve with the Kansas City meeting.

One of the basics of *Kilobaud* is the concept of passing along reader solutions to problems . . . hardware or software. If you figure out how to match a BCD keyboard to a Baudot Teletype to an ASCII I/O, please pass along the info. If you run into a tape reader which doesn't read . . and get it to read . . . let us know. Okay?

ONE DOWN

Wes Schneider, the editor and publisher of *Microtrek*, called to say that his magazine had "merged" with *Personal Computing* and that his plans call for him to move from Cedar Rapids to Pennsylvania and work for *Personal Computing* from there. *Personal Computing* will be continuing as a six times a year publication.

POPULARITY EXPLOSION!



JUPITER II A
6800 System
$795

JUPITER III A
Z80 System
$865

If you thought the quality of a wire-wrapped system was beyond your price range — Take a look at what we have now!

The Jupiter IIA and the Jupiter IIIA Basic computer systems. You get the system module cage with fully assembled backplane, fully assembled plug-in ferro-resonant power supply, front panel kit and your choice of 6800 or Z80 CPU module kit. All less than the price of the two best selling 8080 systems!

Plus you can choose from the fastest growing selection of memories and peripherals available from any manufacturer, like our 2KB EPROM/4KB RAM/serial RS-232 module which can transform your basic computer system into a real star.

And remember, all Wave Mate products meet the highest quality industrial standards, with rugged construction unmatched by anyone! Join the popularity explosion and get yours now! Write or call for more info and your closest Wave Mate authorized distributor.

You get your choice of microprocessors!
And you get wire-wrapped modules too!

Now you have a low cost way to get started into personal computing without sacrificing future growth capability!

Send information on: ☐ Jupiter IIA system
☐ Jupiter IIIA system

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ________ ZIP ________

Wave Mate

WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept 24

The Motorola Way!

... a hobbyist's review of the MEK6800D1

Howard Berenbon
2681 Peterboro
W. Bloomfield MI 48033

The M6800 Design Evaluation Kit from Motorola is, in my opinion, one of the best and least expensive ways for the electronic hobbyist to enter the field of microcomputing. It is available from any Motorola Semiconductor distributor for $149.00. This price includes three I/O (Input/Output) devices; two RAN (random access memory) chips, each containing 128 X 8-bits; one MC6800 Microprocessing Unit; one MCM6830L7 ROM (read only memory) which contains a program called MIKBUG* that allows you to load, display and output data for the M6800; Teletype** and RS232 circuitry for interfacing to a peripheral and a 9¾" x 6" printed circuit board, plated through the holes.

Along with the components, the kit supplies you with two extremely informative manuals and a number of application notes, including the assembly listing for the MIKBUG monitor program. The two manuals are the applications manual and the programming manual. The M6800 Microprocessor Applications Manual is 714 pages in length. It includes a detailed hardware section including circuitry for adding buffers and additional memory for expansion of the M6800 system. Another section details some of the very useful programming techniques of the M6800. It expands on the use of the indexed addressing and relative addressing techniques and describes programming time delays and software design of logic gates. There are some examples provided. They include a short program for use as a time delay and another used for zeroing memory. The zero memory program uses indexed addressing techniques.

The second manual provided by Motorola is the M6800 Programming Manual. It is 300 pages in length. The first part of the book describes the use of an 8K assembler which is available for use with Motorola's Exorciser microcomputer system. There are also sample programs that can be modified to run on most M6800 systems. A third section details each instruction of the M6800 in fine detail for each addressing mode, including a description of the flags which are set or reset upon execution of that instruction. If no flags are affected, then that is also indicated. The number of MPU cycles and the number of program bytes is also given for each operation.

Also provided with the kit is the M6800 Microprocessor Instruction Set Summary Card. This is a 15" x 4¾" card folded into five 3" sections. It includes all of the M6800 assembly instructions and their hexadecimal equivalents for each of the five addressing modes, plus the effect that each operation has on the six condition codes.

Construction and Operation

Construction is fairly straightforward. Motorola does not provide a step-by-step format in their assembly instruction booklet, but they do provide a block diagram of the circuit board with component layout and a detailed circuit diagram of the basic M6800 microcomputer system. This includes all necessary I/O connections. See Fig. 1 for the circuit diagram. There is space provided on the board for adding an additional four MC6810 RAM memory chips. This allows a total of 768 words or bytes of useful memory space without modification to this single-board M6800 system.

The system may be set up for use with a Teletype, which operates at a speed of 10 characters a second, or it may be used with an RS232 format, which operates at a speed of 30 characters a second. The total cost for the M6800 system is approximately $250.00, less the cost of the Teletype or other similar terminal which is necessary for input/output. This includes the price of the design kit and power supplies, plus $50 extra for filling the board with resistors, capacitors, potentiometers, diodes, integrated circuits, IC sockets and four additional RAM chips. To prevent possible heat and static damage to the microcomputer chips, I suggest that IC sockets be used to mount them onto the printed circuit board, and when handling the chips you should be grounded. A con-

*Registered trademark.

ductive wrist strap connected to a ground is suggested.

There are three power supply voltages needed to operate the M6800 system. The main board uses a highly regulated 5 V dc, 1 Amp supply. The Teletype interface requires +12 V dc, 100 mA and -12 V dc, 50 mA, both ±10%. The 5 V dc and the ±12 V dc supplies must be isolated from each other. I built the 5 V dc supply using a LM309, 5 volt, 1 Amp regulator. I used two LM340-12, 12 V dc regulators to design the ±12 V dc power supply. See Fig. 2 for the power supply circuits.

The printed circuit board is mounted vertically in an 86 pin connector (P1). The connector is mounted to a 12" x 22" piece of vectorboard. The 5 V dc power is applied to the PC board through the 86 pin connector. The ±12 V dc supply is connected to the Teletype interface through a 16 pin connector (P3).

There are two critical adjustments that must be made before proper operation of the M6800 system. The bit rate must be set up for either 10 CPS by adjusting potentiometer R23A to 9.1 msec, or for 30 CPS by adjusting potentiometer R23B to 3.3 msec. The clock frequency must be set by adjusting R9 to 450 nsec for $\theta 1$ and adjusting R13 to 470 nsec for $\theta 2$.

Connector P3 interfaces to the Teletype or RS232 type terminal through a 16 pin header. My system uses an ASR-33 Teletype as the terminal. The serial output, TTY common and serial input of my system are wired to an RS232 connector (DB-25S-25 pin female), which is mounted at the rear of my cabinet. A cable coming from the Teletype is wired to an RS232, 25-pin, male connector. The Teletype is wired for a 20 mA, full duplex, current loop, following the wiring diagram in the assembly manual.

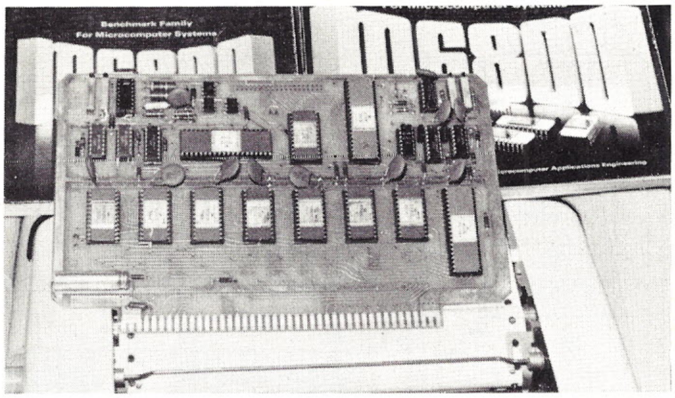
Testing the microcomputer for proper operation consists of turning power on and pressing the reset push-button momentarily. The terminal should respond with a carriage return, line feed an an asterisk which indicates the MIKBUG control program is ready for input. Now you're ready for programming.

Programming

The maximum allowed RAM (random access memory) in the M6800 Design Evaluation Kit is 768 bytes, or words. Since the MCM6830L7 MIKBUG monitor ROM uses one RAM chip (128 bytes) as a stack, this leaves 640 bytes of memory for programming. The memory of the system may be expanded to up to 65K, but this will require additional memory chips and buffering. And 640 bytes is actually more than enough memory space for demonstrating the usefulness of the M6800 microcomputer.

The secret of successful programming is in the MIKBUG monitor ROM. MIKBUG allows you to load data into memory, change the contents of memory and display the contents, punch a paper tape from memory, display the contents of the microprocessor's registers and go to target program.

MIKBUG provides a number of useful subroutines that one may incorporate in programming the M6800. Three of these subroutines are CHARIN, CHAROUT and RETURN TO MIKBUG. CHARIN is accessible by addressing the hexadecimal memory location E078 using the JSR (jump to subroutine) instruction. It allows you to input one character into accumulator A from the terminal each time the subroutine is called. CHAROUT is accessible by addressing the memory location E075. It will output one character from accumulator A to the terminal. RETURN TO MIKBUG allows control returned to MIKBUG. It is accessible by addressing the memory location E0E3.



Before going on to the programming example, here is a description of the features of MIKBUG. To load data one simply types the address of the beginning hexadecimal byte that you wish to start at. This is done by first typing an M at the terminal. MIKBUG will then type a space. Enter the desired hexadecimal address of the memory location to be opened, and the terminal will respond with a space and the contents of that memory location. The data may be changed by typing a space and then entering the desired data in hexadecimal. The terminal will respond with a carriage return, line feed and print the next consecutive memory address along with its contents. To exit this function, type a space and a carriage return. The terminal will respond with an asterisk indicating control has been returned to MIKBUG. Any number of consecutive memory locations may be displayed when using this function by simply typing any character at the terminal except a space or a carriage return, and MIKBUG will print the next address and its contents.

To display the contents of the MPU registers, type an R at the terminal, and MIKBUG will respond with the contents of registers of the M6800. Starting from the left, the first location displayed contains the condition codes, next the contents of accumulator B, the contents of accumulator A, the index register, the program counter and the stack pointer. The registers are stored on the user's stack.

To execute a program, the program counter must be initialized before a program can be run. Initialize the program counter to the address of the first byte of your program. This is done by examining memory locations on the stack at A048 and A049, which contain the contents of the program counter. Change the contents of these two bytes to the address of the first byte in your program. To execute your program type the letter G at the terminal. This is the GO command. The MPU takes the address in the pro-

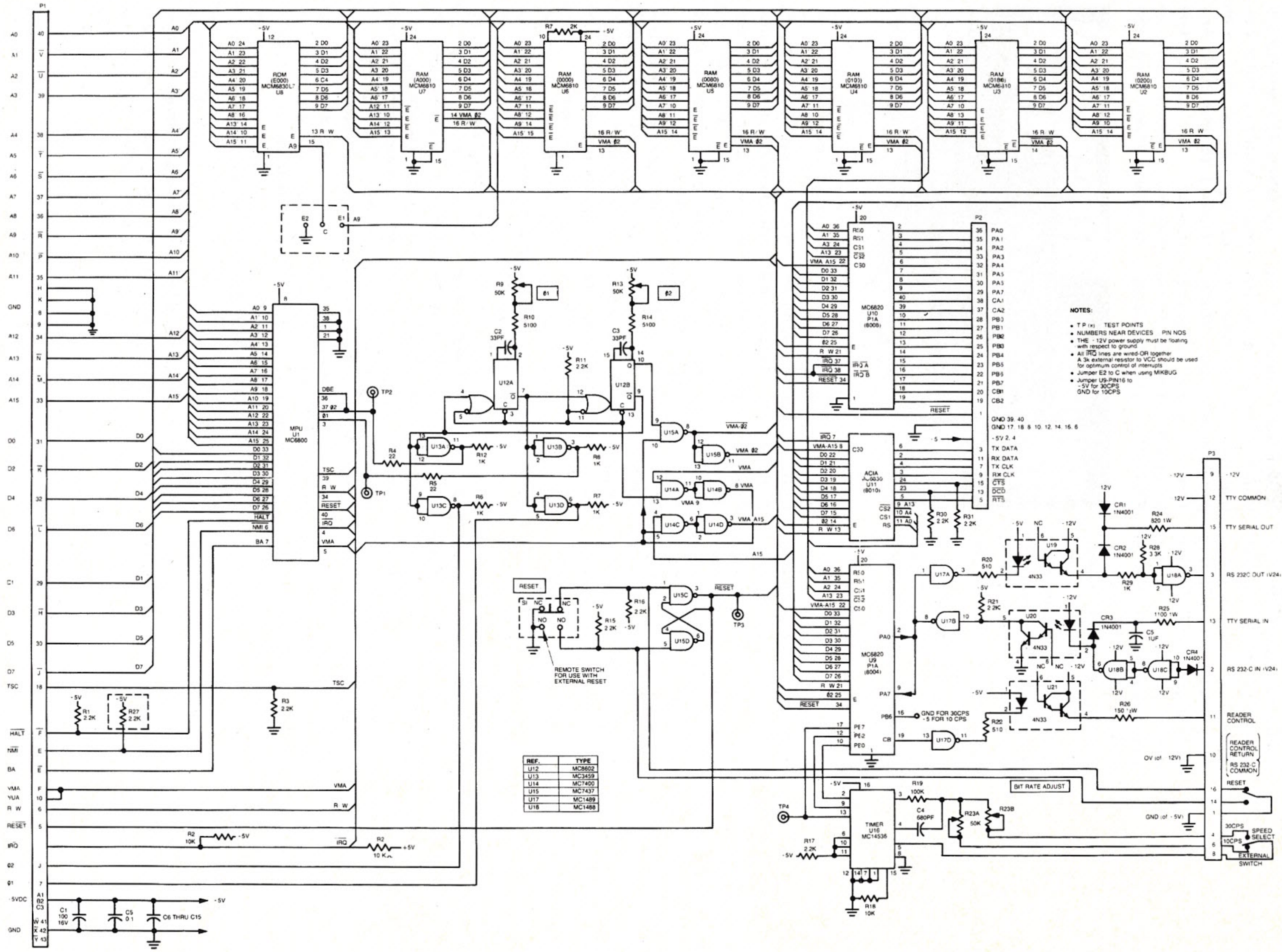


Fig. 1. Interconnect schematic diagram for the MEK6800D1. Reprinted with permission from Motorola, Inc.

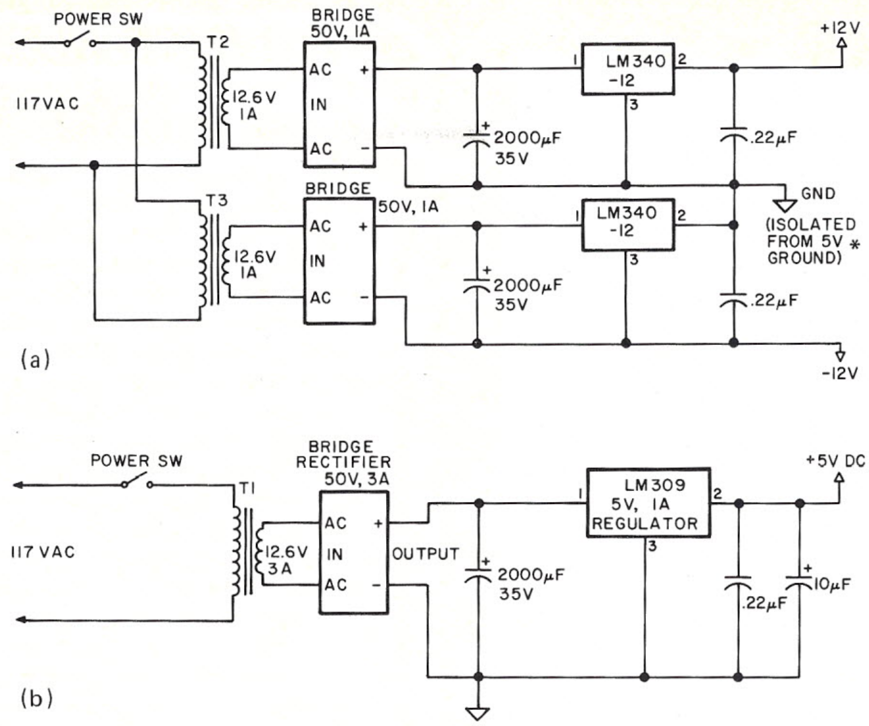
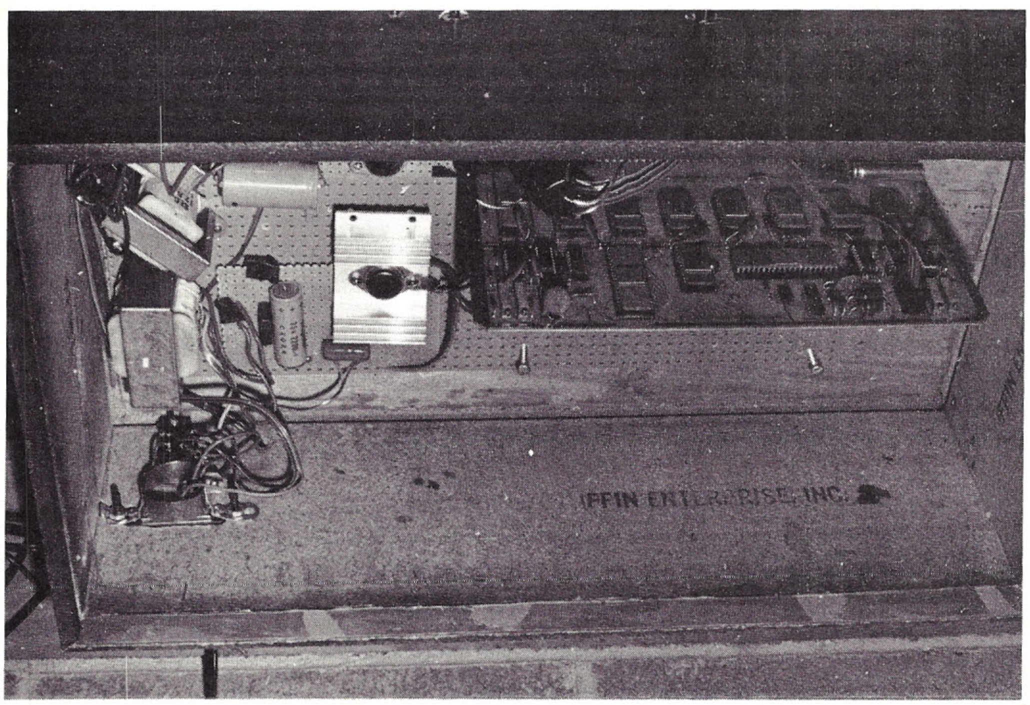


Fig. 2. (a) Schematic of 5 V dc power supply and (b) ±12 V dc power supply which author built to power unit.

gram counter and begins execution at that address. To exit the program and return to MIKBUG, just press the RESET button.

A punched paper tape may be made of your program or of any part of your memory by first initializing the stack at A002 and A003 to the beginning address of your program and then entering at locations A004 and A005 the ending address of your program or block of memory. Now feed the paper tape into the punch. Return to MIKBUG by typing a space and a carriage return or by depressing the RESET button. Type the letter P at the terminal. This will turn on the paper tape punch and punch a paper tape of your program in the form accepted by your M6800 system. This program can be reloaded into your system by using the memory loader function of MIKBUG. Place the punched paper tape into the paper tape reader of your terminal. Return the system to MIKBUG control. Now type the letter L at the terminal. The tape reader will be turned on and load the contents of the paper tape into your system's memory ready for use. To run the program, initialize the program counter as described. Return to MIKBUG and type the G command.

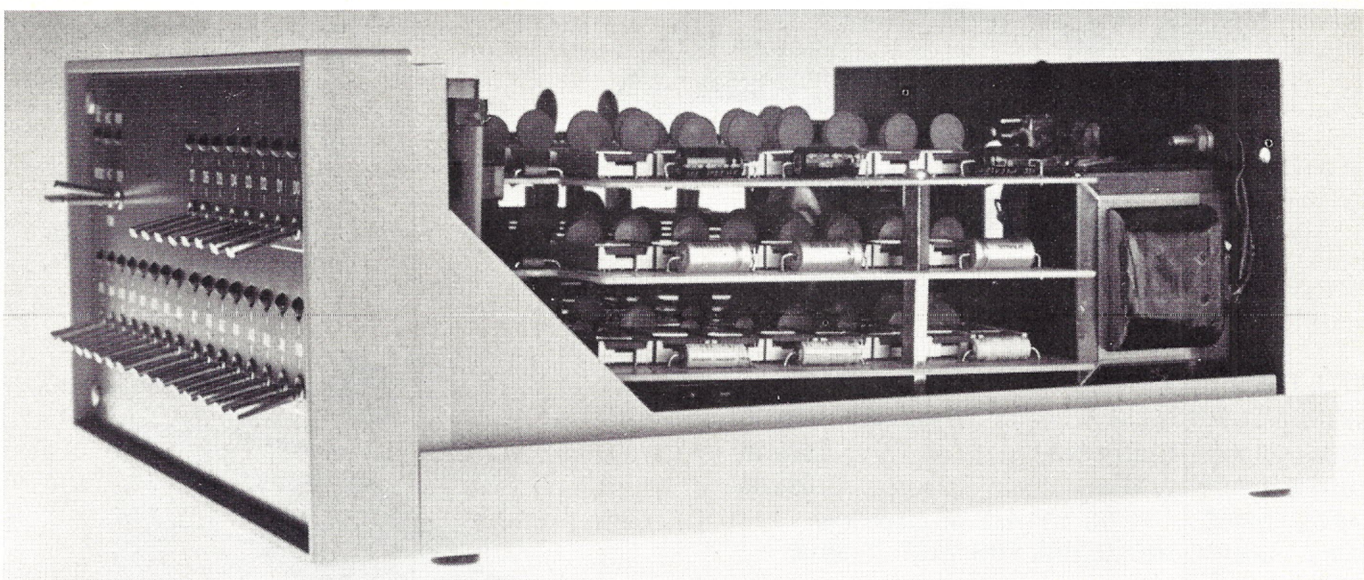
The SWI (Software Interrupt) instruction is an interesting feature of the M6800 and is used in debugging programs. When placed in the body of a program, the program will run until the MPU encounters the SWI instruction. Then the contents of the microprocessor's registers will be printed. These values may be checked against known or calculated values to determine if your program is functioning properly. It can also be used to find problem areas in your programs by replacing one instruction in your program sequence with the SWI. Run the program. Continue to replace one instruction with the SWI until the registers fail to be displayed. Your problem will lie near the last area in which you replaced an instruction with the SWI. ■



Anthony R. Curtis
246 Woodland Dr.
State College PA 16801

Let's Hear It for the 680b!

... an easy building project



Nobody could have been more surprised than I. The Mits Altair 680 looms large in magazine ads. I expected boxes of parts and truck loads of trouble in soldering the brute together. My first surprise was the tiny package delivered by the mailman. The entire 680 came packed into the space of two shoe boxes. The second surprise was in the building. The tiny Altair is a darned simple kit to build.

The 680 is, of course, a complete central processor around which a fine system can be built. Yet, it has fewer parts than the Heath HW-8 QRP amateur transceiver for example. To build the Altair, all you do is drop 56 integrated circuits into clearly marked holes on two printed circuit boards and solder them in. A couple dozen toggle switches and LEDs plus a big bag of resistors, capacitors and miscellaneous parts fill the boards.

The whole computer can be built in two days. I sat down at the bench on a Tuesday morning and had the machine running Wednesday night. And that included plenty of lunch, coffee and TV breaks plus *no* working from 5 p.m. Tuesday to 9 a.m. Wednesday. This kit is not complicated.

The 680 is less than one third the size of and costs less than half the price of Altair's big 8800 microprocessor. For $466 you get a kit of what I counted at 451 parts, which compares with my count of 518 parts in the also-easy-to-assemble HW-8. You get an easy-to-understand assembly manual with the 680 but not enough instruction in how to make the box do tricks after you have built it.

Among the goodies inside the 680 mailing carton you'll find two predrilled fiberglass printed circuit boards. The boards have a hefty feel and a commercial-grade look with PC lands on both sides and holes plated through so you only have to solder one side. Each board has one side silk-screened with parts locations clearly marked.

Also in the box are three fat plastic bags of small parts, a two-inch thick ring binder of instructions, and a solid blue and gray box to house your finished computer. A careful count of materials shipped showed that Mits had included everything listed in the parts list. I didn't have to call Albuquerque back for missing numbers.

The only tools needed to build the 680 are a 25-30 watt soldering iron with chisel tip (I used the Ungar model 776 with matching tip number 7155), a pair of needle-nose pliers, a pair of diagonal cutters, a standard-blade screwdriver and a Philips-blade screwdriver.

I think Mits assembly instructions are as easy to understand as those provided by the better-known kit seller, Heath Company. And I've known some terrible bumblefingers to complete Heathkits successfully. The comparison with the QRP transceiver is important since I consider the HW-8 only *medium* hard to construct. If you *never* have wired anything before, I would suggest trying something extremely simple such as a Heathkit code practice oscillator or the like to get a feel for use of soldering iron, pliers, cutters and the like. On the other hand, if you have built something complex like a Heath HW-101 or even an SB-104, then the Altair 680 will be child's play for you.

Whether you are a beginner or not, take it slow and easy while wiring the 680. There are some very dangerous pitfalls to be avoided:

1. Don't use a soldering iron of more than 25-30 watts since higher power irons and guns will lift the metal lands off the printed circuit board and ruin the circuit.

2. Don't heat individual connections too long with your iron as too much heat will ruin some parts as you solder them in.

3. Be very careful to heat individual connections only long enough to get a minimum amount of solder to flow around it since the lines printed on the boards are tiny and close together and any solder which would accidently flow across from one line to another, forming an electrical bridge where none was intended, can blow up your microprocessor.

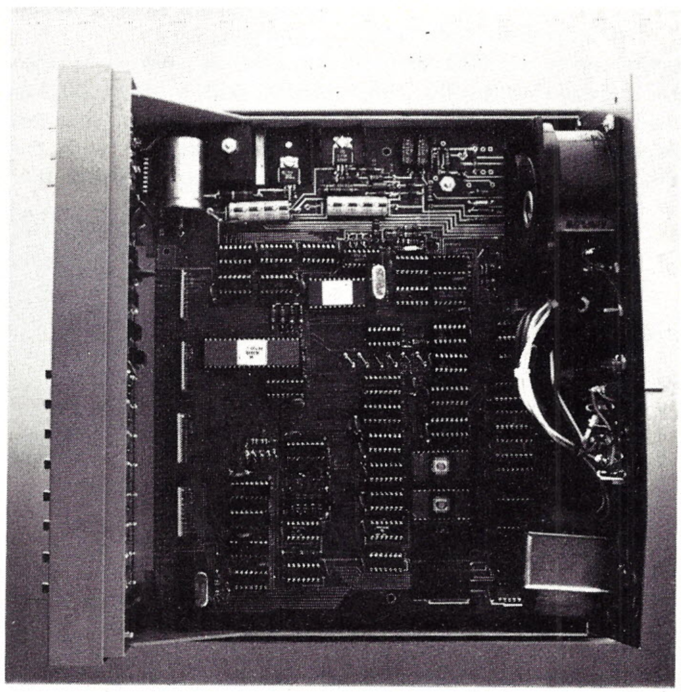
4. Use plenty of bright light to see your work and guard closely against forming bridges.

5. Handle the CMOS integrated-circuit chips cautiously to avoid zapping them.

TTL chips, which carry numbers in the 7400 series, require no special handling. But, watch that CMOS. It requires careful handling so static electricity, around your body and elsewhere in the vicinity of where you are handling and soldering the chip, doesn't arc across the chip and destroy its innards. If you do blitz a CMOS chip, you may not know it until you have finished wiring up the kit and have it buttoned up in its cabinet. Then when you try to make the machine compute and it won't, that might be the reason.

Another cause of your 680 not working right at first could be one of those elusive solder bridges on the PC board. Those fine lines printed on the boards are very close together and it's easy to mistakenly bridge the gap.

Speaking of not working right the first time, mine suffered too. It turned out I was not careful enough in placing several jumper wires inside the 680. The wires tell the 680 how to think and communicate with the outside world. I mistakenly strapped a jumper so the 680 was searching for an outboard terminal on which to display answers to problems I had generated. I wanted to see the answers among the front panel LED lights but the machine wouldn't put them there. Opening up the case and reviewing placement of the various jumpers, I found I had misplaced the jumper which indicates to the microprocessor whether or not an outboard terminal is connected.



There are a number of such jumpers to be soldered inside the 680 cabinet on the main PC board. I would have preferred some arrangement of molex pins or other type of connectors instead of soldering.

The internal jumpers do such jobs as tell the machine what baud rate, from 50-9600, it is expected to use in talking with its peripheral devices; where to start looking in memory for programs; whether or not an old fashioned five-level Baudot Teletypewriter* is being used to put information in and take answers out of the computer; and whether or not any outboard terminal is being used with the microprocessor.

The holes for these jumpers are marked on the circuit board and the instruction book explains how to wire for your choices. But, I found, it is easy to make a mistake and get the end of one jumper in a wrong hole. Check your work carefully.

The 680 is only 11 inches wide by 11 inches deep by 5 inches high, but don't think it is weak because it is small. This is a complete general-purpose computer capable of being programmed from front

---

*Registered trademark

panel switches or interfacing with video-display terminals and Teletype machines. The basic 680 comes with 1K of RAM memory on the main board in the form of eight 2102 chips. Each of these holds 1,024 bits of information. Eight side-by-side provide a means of holding 1,024 eight-bit words or bytes. You talk to this computer in such bytes.

The main circuit board is horizontal in the bottom of the case. There is room for three more horizontal boards above the main board for such things as additional memory capacity. Mits sells a board with 16K of memory to plug in here for $685. And, for that figure, they throw in the BASIC language to make talking to the blue and gray box easier.

The second printed circuit board inside the 680 cabinet is mounted vertically, immediately behind the front panel. The board holds 12 of the IC chips, 27 toggle switches and 27 LEDs. Controlling the front panel display, this board plugs directly into the horizontal main circuit board via a 100-pin edge connector on the front lip of that main board.

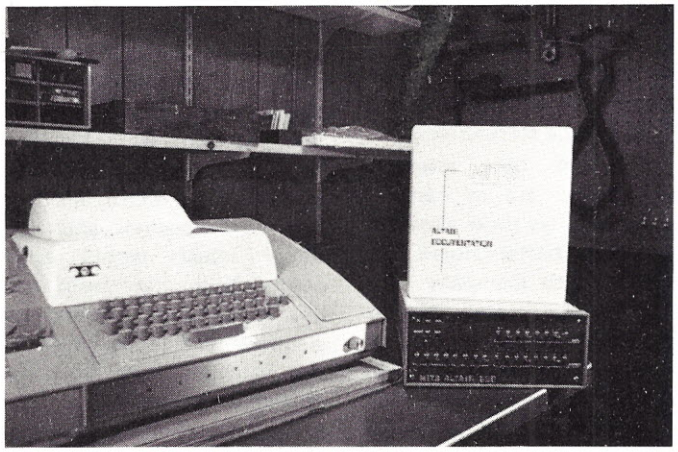
By the way, 15 of the IC chips supplied by Mits come with sockets. These holders are soldered into the printed circuit boards and the ICs are plugged into the sockets for easy removal. All other chips are soldered directly into the holes in the PC boards.

The 680 seems an ideal way to get started in microprocessors. Using its front panel switches and lights, you can learn how to place programs and data in memory and withdraw them later. You can run simple programs requiring not more than 1K of memory space and familiarize yourself with how such machines work.

Later, as your knowledge grows, add a video display terminal (VDT) so you can talk more quickly with the machine. Add 4K, 8K or 16K of memory plus a high level language such as BASIC and you'll have all the system you now can dream of. The Altair is a popular rig and the Motorola 6800 microprocessor chip at the heart of the 680 is well known and widely used. Many other manufacturers sell memory boards, cassette interfaces, VDTs, teletypewriters and other gadgets to plug into the 680.

Despite the thick notebook of instructions supplied by Mits, I found info lacking in the area of programming. Being a complete novice in the computer hobby, I needed elementary instruction in how to make the machine add two and three and get five.

You can, of course, buy the 680 wired and tested for $625. But why not save $159? The 680 is easy to build and quickly placed in operation. Once you have the experience of wiring the 680, think of all the accessory boards you can build to expand your system and save money as you go. ■



If You're Still Playing Games It's Because You Haven't Seen Our Software Library

This Library is the most comprehensive work of its kind to date. There are other software books on the market but they are dedicated to computer games. The intention of this work is to allow the average individual the capability to easily perform useful and productive tasks with a computer. All of the programs contained within this Library have been thoroughly tested and executed on several systems. Included with each program is a description of the program, a list of potential users, instructions for execution and possible limitations that may arise when running it on various systems. Listed in the limitation section is the amount of memory that is required to store and execute the program.

Each program's source code is listed in full detail. These source code listings are not reduced in size but are shown full size for increased readability. Almost every program is self instructing and prompts the user with all required running data. Immediately following the source code listing for most of the programs is a sample executed run of the program.

This Library is destined to become one of the reference bibles for the small computer field, due to its versatility and uniqueness and the ease of operation of the programs it contains. These volumes are deductible as a business expense when purchased by a company. Send your remittance for prompt delivery, while supplies last. Volume discounts are available to qualified dealers.

The entire Library is 1000 pages long, chock full of program source code, instructions, conversions, memory requirements, examples and much more. ALL are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers available for 8080 & 6800 under $10 elsewhere.

This LIBRARY is a complete do it yourself kit. Knowledge of programming not required. EASY to read and USE. Written in compatible BASIC immediately executable in ANY computer with at least 4K, NO other peripherals needed.

VOLUME ONE

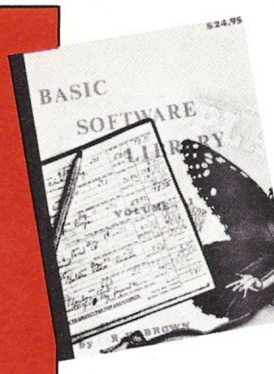
Part 1

BOOKKEEPING
Bond
Building
Compound
Cyclic
Decision 1
Decision 2
Depreciation
Efficient
Flow
Installment
Interest
Investments
Mortgage
Optimize
Order
Pert Tree
Rate
Return 1
Return 2
Schedule 1

Part 2

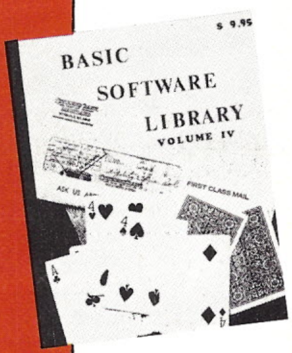
GAMES
Animals Four
Astronaut
Bagel
Bio Cycle
Cannons
Checkers
Craps
Dogfight
Golf
Judy
Line Up
Pony
Roulette
Sky Diver
Tank
Teach Me

PICTURES
A. Newman
J.F.K.
Linus
Ms. Santa
Nixon
Noel Noel
Nude
Peace
Policeman
Santa's Sleigh
Snoopy
Virgin



VOLUME FOUR

Bingo
Bonds
Bull
Enterprise
Football
Funds 1
Funds 2
Go-Moku
Jack
Life
Loans
Mazes
Poker
Popul
Profits
Qubic
Rates
Retire
Savings
SBA
Tic-Tac-Toe



VOLUME TWO

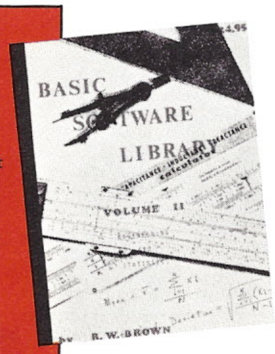
Part 3

MATH & ENGINEERING
Beam
Conv.
Filter
Fit
Integration 1
Integration 2
Intensity
Lola
Macro
Max. Min.
Navaid
Optical
Planet
PSD
Rand 1
Rand 2
Solve
Sphere Trian
Stars
Track
Triangle
Variable
Vector

Part 4

PLOTTING & STAT
Binomial
Chi-Sq.
Coeff
Confidence 1
Confidence 2
Correlations
Curve
Differences
Dual Plot
Exp-Distri
Least Squares
Paired
Plot
Plotpts
Polynomial Fit
Regression
Stat 1
Stat 2
T-Distribution
Unpaired
Variance 1
Variance 2
XY

APPENDIX A
BASIC
STATEMENT DEF



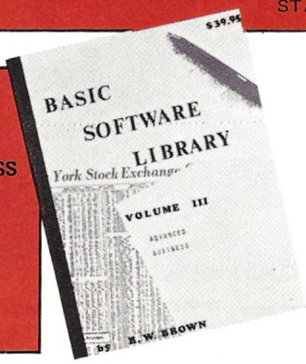
VOLUME I & II – $24.95 each
VOLUME III – $39.95 each
VOLUME IV – $9.95 each

Add $1.50 per volume for postage and handling. 10% discounts on purchases of any three (3) volumes. Money orders and bank card orders shipped same day. C.O.D. and checks take longer.

VOLUME THREE

Part 5

ADVANCED BUSINESS
Billing
Inventory
Payroll
Risk
Schedule 2
Shipping
Stocks
Switch



SCIENTIFIC RESEARCH

1712-K FARMINGTON COURT
CROFTON MD 21114

master charge THE INTERBANK CARD

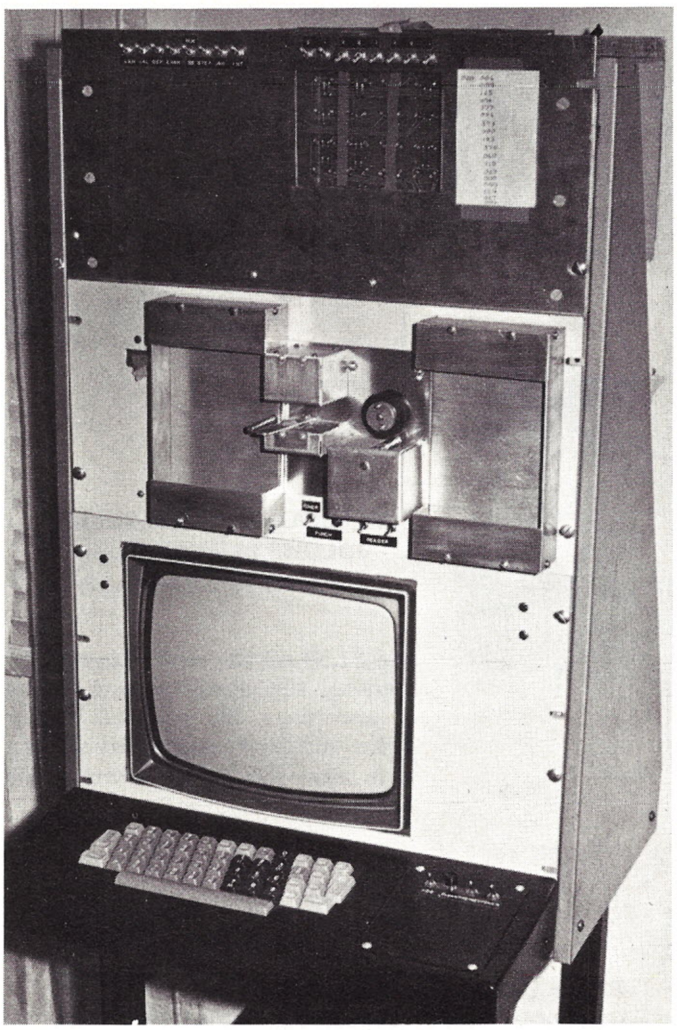
your BANKAMERICARD welcome here

Phone Orders call (800) 638-9194
Information and Maryland Residents Call (301)-721-1148

Dr. Doug Hogg
36 Calle Capistrano
Santa Barbara CA 93105

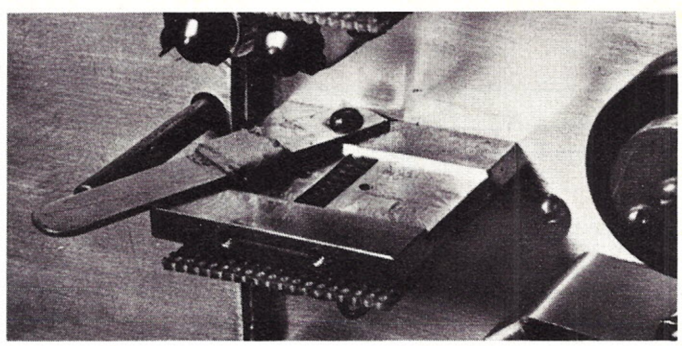
The Paper Taper Caper

... build your own tape reader



Home brew high-speed paper tape reader rack mounted with a Mark-8, TV Typewriter, paper tape punch, and keyboard.

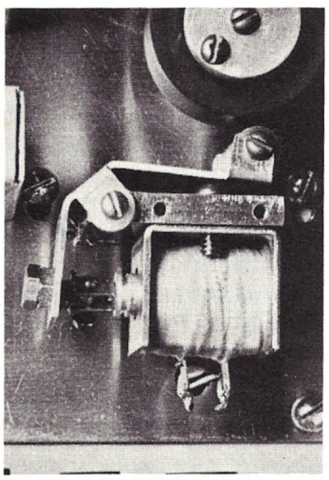
Doug Hogg is one of the earlybirds to the computer hobby movement. He got started with the original Mark-8 and is in fact still going strong with it. With the vast amounts of software running around in paper tape form, his article on constructing a good reader should be quite valuable. The article is written with the 8008 and 8080 in mind, but there's no reason why it won't apply to any other processor. – John.



Phototransistor array (PTA) in the tape guide mount. The two holes are for the mounting screws which hold the PTA in place. The lever in back is provided to hold the tape down while it passes over the PTA.

One of the most pressing needs of the computer hobbyist is for a high speed input device. Audio cassette interfaces are nice but generally slow with even the "fast" units seldom running more than 100 characters per second. The paper tape reader described here will run at speeds of up to 600 or so characters per second. This means you can load 12K BASIC in 20 seconds as compared with six minutes for the Kansas City standard cassette interface. Paper tape has the advantages of being easy to handle, convenient storage for any length program, and you can actually *see* the data on the tape. On the negative side, paper tape *punches* are expensive, slow (comparable to cassettes), and noisy but the ratio of input to output in the I/O system is large enough to minimize this nuisance.

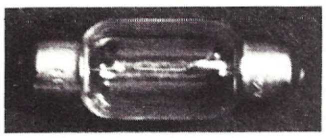
Even more important than the speed and convenience of the paper tape is its enormous flexibility. The reader can read 5, 6, 7, and 8 level tapes in *any* format. The Tarbell cassette interface cannot talk to the Kansas City standard or many others. Of course, if you do not plan to use anyone else's software, then this is not important. Unfortunately there is no standard format with paper tape either, but this is now a software interpretation matter rather than a severe (and perhaps expensive) hardware problem.



Solenoid and pinch roller assembly. Two screw holes are provided in the front to mount the cover.

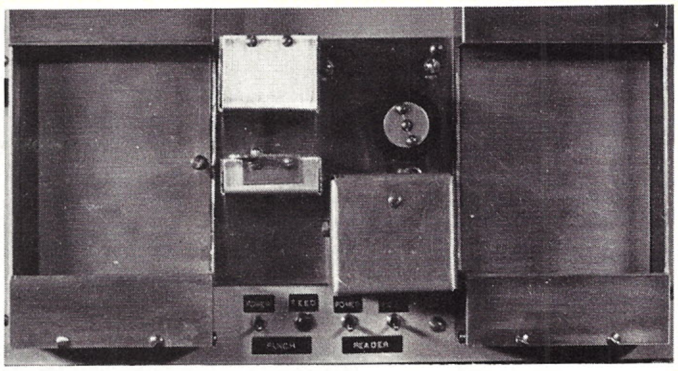
The following sections discuss the mechanical and electrical aspects separately. Although this unit can, with minor variations, be built with only hand tools, those people with no shop facilities can still build the unit without a motor for manual operation; this is not particularly recommended, but it can produce a serviceable unit. With manual units, there is generally some sacrifice in the error rate. Later sections discuss various construction options, interfacing, and software.

The basic electrical schematic is shown in Fig. 1. The phototransistor array (PTA) is the heart of the unit. It consists of nine encapsulated phototransistors aligned and spaced on 0.100" centers. For most units the pull up resistors on the gate inputs are not necessary, however they can be used to compensate for the different sensitivites of the individual phototransistors in the package.



Number 211 bulb used for light source in reader. Note the long filament.

The light source is a number 211 bulb shown in the photo. This bulb, designed for automobile door lights, is well suited to our purposes because of its long filament and availability. With a separation of about two inches between the light and the paper tape, you can run this 12V bulb on 6V. At such a voltage, its life should be essentially infinite. The socket is made from a clip-in fuseholder or you can purchase a ready made socket — if you can find one. If not, the fuse clip trick works well and is shown in the photo of the lamp mounting.



The completed unit. The slot on the left is for the paper out from a paper tape punch.

Mechanical Construction

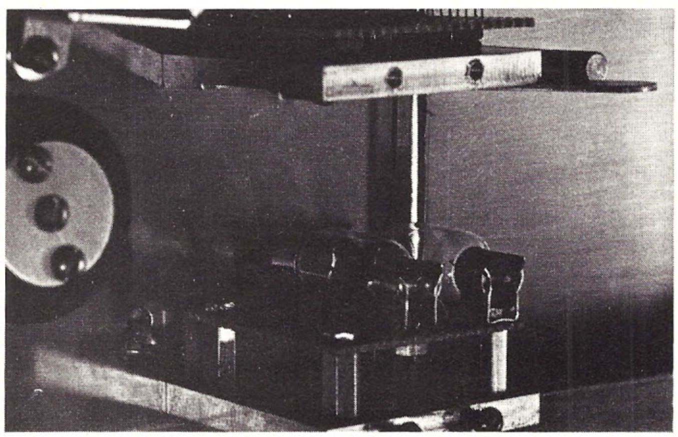
The photo (above) shows the front view of a completed unit. The paper tape moves from the supply box on the left, under the guide post, along the tape guide (which is also the mount for the phototransistor array), between the drive wheel from the motor and the pinch roller on the solenoid, and finally into the collection box on the right hand side.

Before discussing the motor mounting, it might be a good idea to discuss the motor. The motor I used (surplus) was a shaded pole geared unit with a ¼" shaft turning 310 rpm. This, coupled with a rubber rimmed drive wheel (also surplus), happened to give a tape speed of 50 characters per second — slightly slow but in the right ballpark. Before anyone goes out and buys a fancy motor remember that the geared units are noisy and that a high-speed, small shaft motor is equivalent to a low-speed motor with a large drive wheel. For instance, a motor which operates at 3600 rpm and has a ¼" shaft gives a speed of 470 characters per second. So whatever motor is handy will probably be all right. However, with a steel shaft you should use a rubber pinch roller. Again, the most important requirement is that the motor be quiet.

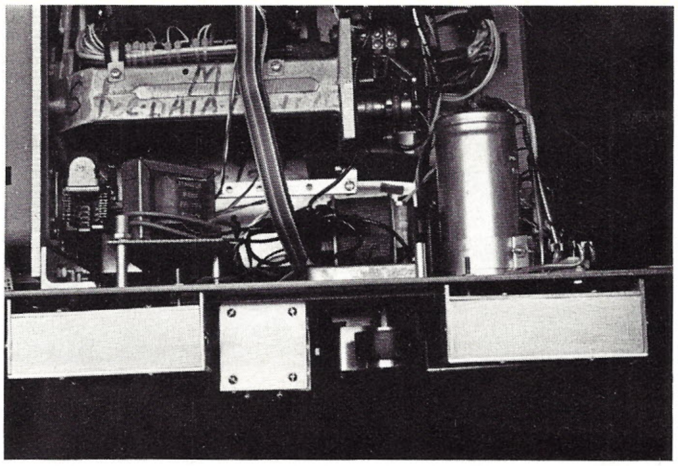
The lower side of the motor shaft should be slightly below (1/8" to 1/4") the edge of the tape guide and electronics assembly — refer to the front view photo.

When mounting the motor be careful not to strain the frame, as this distorts the motor and it will not run well. It is best to tighten the bolts with motor running, using the sound as a guide.

The next part, the pinch roller and solenoid assembly, (see photo) presents more options. My configuration



Lamp mounting. The holder is made from a fuse clip. The bulb on the left is a spare.



Top view of the paper tape reader assembly. Note that the supply and collection boxes are spaced out from the panel. The punch is behind the reader.

was chosen partly to satisfy my urge to see the computer actually do something – in this case start and stop the motion of the paper tape. Construction of the pinch roller holder and pivot was made necessary since the most common solenoids are pull type while the unit needs a push type. The pivot, pinch roller, and solenoid mount dimensions are shown in Fig. 2. This pivot was made from 1/16" aluminum, the mount from ¼" aluminum and the pinch roller from a 3/8" brass rod mounted on a 6-32 screw 1½" long. (Study the photo and Fig. 2 and associate each with the three parts discussed.)

Fig. 3 shows alternative simpler ways to hold the pinch roller using either push or pull type solenoids. In this case, the pinch roller mount can easily be made by hand from wood. In fact, even the pinch roller can be a piece of ½" dowel. These methods are also satisfactory, without the solenoids, if you want only manual stop and start.

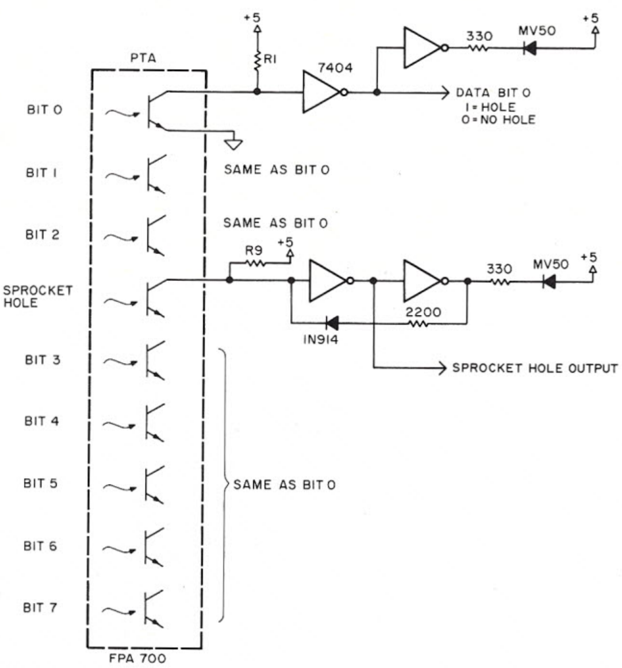


Fig. 1. Electrical schematic for the paper tape reader. All eight data bits are identical.

We come next to the only piece of the construction that requires careful work. The holder for the phototransistor array (see photo) requires a mill. (For those who do not have access to a mill, we will present alternative designs later.) The basic features of the piece are a channel to guide the paper tape and an accurately positioned slot for the phototransistor array (see Fig. 4). Since the sprocket hold in the paper tape is about 0.040" diameter, if the phototransistor array is misaligned by more than about 0.020", the unit will not read paper tapes properly. You have the choice of either milling an accurate slot or adjusting the position of the phototransistor array later. In addition, the slot must be perpendicular to the paper tape guide so that all the holes read belong to the same character.

My holder was machined from a 2" square block of ¼" aluminum as follows: 1) the paper tape channel was milled to a depth of about 0.05" and polished (see photo and Fig. 4. Fig. 4 illustrates a deeper, ¼" slot.) 2) A slot across the paper tape channel ¼" wide by one inch long was milled to hold the phototransistor array. 3) A slot 3/8" wide by 1.5" long was milled along the same line as the previous slot but only to a depth of 1/16". The phototransistor array sits in this slot. 4) The corners of the first slot were squared up with a file. 5) Two 4-40 holes were tapped near the slots so that the shoulder of the 4-40 screw could be used to hold the phototransistor array in place. 6) Finally, the piece was carefully polished so that paper tape would move *smoothly* through the channel; rough spots destroy tape very quickly.

If you have only hand tools, the phototransistor array holder can be made in the following way using either wood or metal. The basic construction is shown in Fig. 5. First the slot is cut in the base for the phototransistor array. The top pieces are then bolted in place to form the 1" wide tape channel. Adjust the position of the first piece to obtain proper alignment before attaching the second piece. After the pieces are mounted, drill the 1/16" hole for the tape guide rods. This method requires more care for alignment but works as well as the one piece holder. Even simpler holders are possible if you make notched guide posts to place before and after the phototransistor array which can then be held by any stable mount. In any of these mounting schemes, be sure there is a small amount of clearance (a few thousandths of an inch) between the paper tape and the surface of the phototransistor array because the tape can scratch the plastic surface of the phototransistors.

Now we need a method of holding the paper tape down as it passes by the phototransistor array. From the photos you can see that I used a simple lever, with a soft material on the underside, which swings over the tape channel. A better way (as shown in Figs. 4 and 5) is to start with a thicker piece of material for the phototransistor holder (½" instead of ¼"), mill the tape channel to 0.30", and drill two transverse holes 1/16" diameter so that two rods can be inserted across the tape channel to hold down the paper tape as it passed the read head. For loading the paper tape the rods are withdrawn. It is a good idea to hold the tape down on both sides of the phototransistor array, particularly if fan fold tape is used as it tends to lift up at the folds.

The lamp and mounting are shown in the photo. The second bulb is a spare and a totem of sorts. A variation of Murphy's Law says that the life of the bulb in use will be doubled or tripled by having

a spare bulb in a very convenient place (the inverse is true if one is not available).

Once all these pieces are made the reader can be assembled. My reader was assembled on a subpanel (7" x 5") which is bolted to an 8 ¾" x 19" relay rack panel. (Refer to Fig. 6 and the front panel photo.) Before the final assembly, check to see which direction your motor turns because the tape must be pulled through the reader. If you use roll tape, mount a post for the supply roll and use a large cardboard box to collect the tape. Or you can always wind it up on the motor drive shaft. With fan tape you need to make two boxes for the tape (see Fig. 7). The inside dimensions I used are 7½" tall, 1¼" deep, and 4" wide. Although my boxes are made of aluminum, plexiglass is much easier to work with and it gives a very nice looking finished box. A half strip of corrugated cardboard in the bottom of the collection box helps the tape fold smoothly. When mounting the supply tray (see top view photo), align the back side of the tray slightly behind the back edge of the paper tape channel. An additional guide post to the right of the supply tray was found to be helpful in keeping the paper tape in the guide channel. The paper tape goes under this guide.

Additions

In the course of describing the construction, I have indicated some of the changes and improvements that in retrospect seem warranted. The most useful addition not described before would seem to be a brake so the reader can be stopped on character.

How hard this is to accomplish depends on your final tape reader speed. In my case, the reader takes five characters to stop (about 20 msec). This could be improved considerably by attaching a spring to the solenoid, as gravity is now used to relax it. For moderate tape speeds (less than about 100 characters per second) this may be sufficient. For faster rates, some sort of dynamic

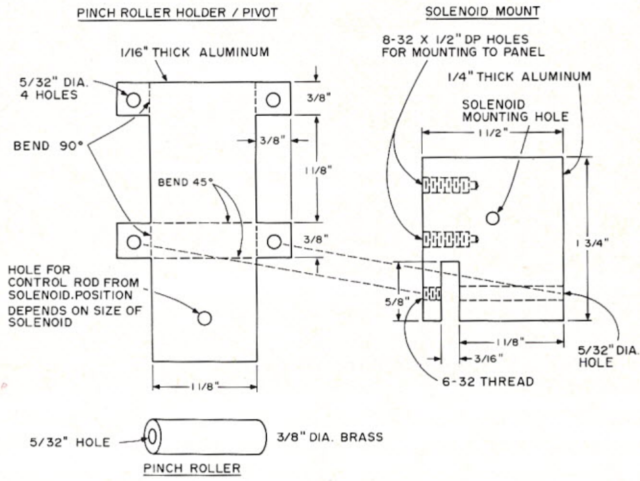


Fig. 2. Dimensions for pinch roller, pinch roller holder, and solenoid mount.

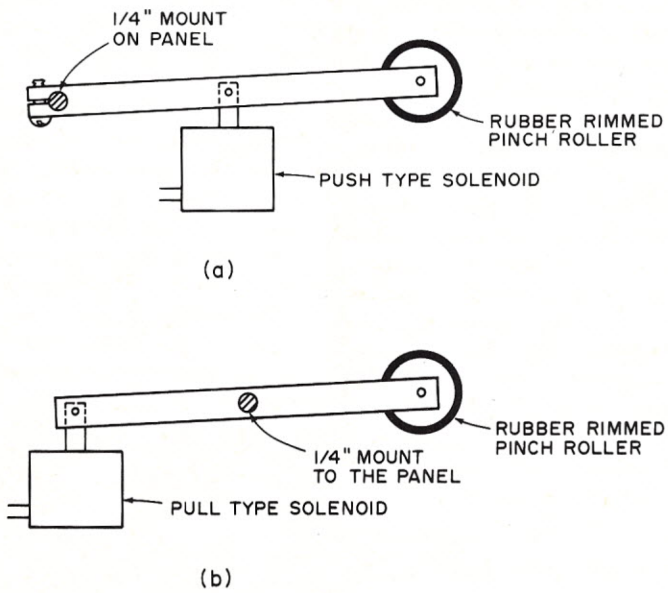


Fig. 3. Alternate methods of attaching the solenoid to the pinch roller with a push type solenoid (a) and a pull type solenoid (b).

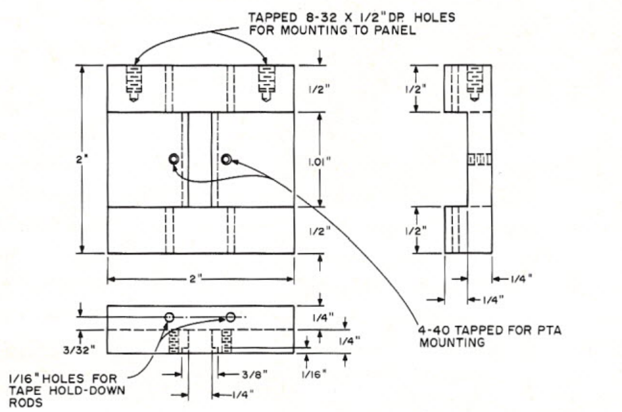


Fig. 4. Phototransistor array (PTA) holder and paper tape guide channel.

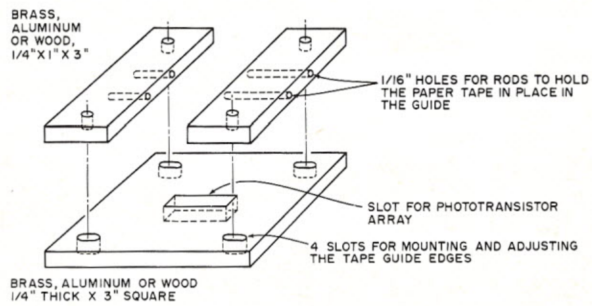


Fig. 5. Alternate method of construction the mount for the phototransistor array and the tape guide.

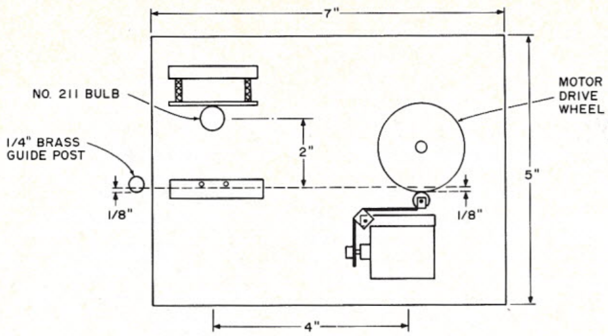


Fig. 6. Dimensions of the subpanel and placement of components.

brake is necessary. At 500 characters per second the tape must stop in less than 1 msec. Since at these speeds it is difficult to move a brake very far, a pressure mechanism must be used. The brake should be installed on the opposite side of the phototransistor array from the motor. One method for the construction of the brake is shown in Fig. 8. A coil of wire, or a solenoid without the plunger, has an iron piece spring mounted above it. When the tape is to be stopped, the brake solenoid is energized and the tape is pinched between the magnet and the top of the solenoid. If the form for the phototransistor array is enlarged the magnet can be used for part of the tape guiding mechanism. This sort of brake is used in commercial readers so it should be very effective. If your coil or solenoid is not a coninuous duty type, then the unit can be pulsed for a few milliseconds, just long enough for the tape to be fully stopped. Two versions of the control circuitry (one continuous and one pulsed) are shown in Fig. 9.

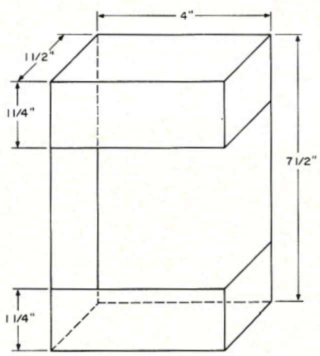


Fig. 7. Paper tape supply box. The collection box is a mirror image of the supply box.

8008 Interfacing With Interrupts

Now that you have assembled all of this it does you no good unless it can talk to your computer. The first interface described is primarily for an 8008 system but it can be used on any interrupt driven system. The second interface is for the more conventional polled operation.

For interfacing to a machine with a front panel interrupt instruction port (such as the Mark-8, not Altair) the eight data lines go to an input port and the strobe line goes to the interrupt input. With my Mark-8, the interrupt expansion circuit shown in Fig. 10 was used because the TVT, punch and keyboard also use interrupts. If the paper tape reader is your only interrupt input (very unlikely), then you do not have to worry. If it is not, more work is required since the reader does not stop on character. My unit takes about 0.5" (5 characters) to stop and thus generates five interrupts past the end of the desired data. To eliminate this deficiency, we have the computer enable the interrupt before reading and disable it when it is through. This is done with the circuit shown in Fig. 11. The instructions for using this are:

```
              8008
LAI           006
004           004
OUT CTRL      125
.
.
.
.
.
HALT          000
IN PTR        103
.
.
.
.
.
LAI           006
000           000
OUT CTRL      125
```

The first three instructions enable the interrupt and the last three disable it. In this case the computer must know when it is through, that is, when all the data you wanted entered has been entered. One way to do this is described below after the solenoid control circuitry.

As long as we are now sending a control signal to the tape reader at the beginning and end of the input procedures, we may as well have the computer actually start and stop the reader (see circuitry in Fig. 12). Besides allowing you to control things from the keyboard, this will surely impress your friends. The optical isolator is not optional. It is there to keep the grounds separate, since mechanical things are electrically noisy. Even with just the ground wire connected between the computer and solenoid control assembly, operating the tape advance switch will generate garbage in the computer.

This circuit is controlled exactly as the interrupt disable circuit except that we use an additional bit on the control output port. The software is the same as above except that instead of loading 004 in A we load 006. The complete tape input routine for the 8008 is shown in Fig. 13.

This last circuit completes the paper tape reader. Now on to the software. The format I have been using on my paper tapes is:

Dummy character
Starting high address
Starting low address
Ending high address
Ending low address
Data

A common variation on this is to have the number of characters instead of the ending address. In the above case, the data can be the entire program or the tape can be organized into blocks (generally 256 characters or less). The simple program in Fig. 13 considers all the data as one block although it can be used as a subroutine in a larger program for using data blocks. Again, the program in Fig. 13 is for an 8008 with the reader input at port 103 and the output control port 125.

8080 Software Interfacing

Fig. 14 shows an 8080 program used to load paper

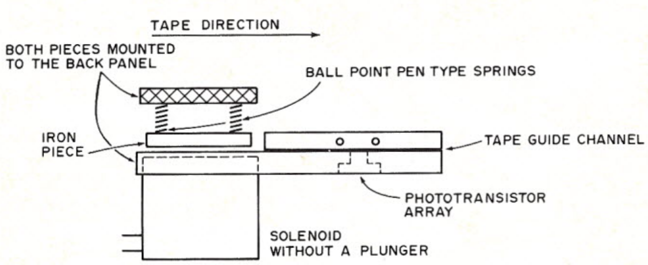


Fig. 8. Brake assembly. The iron piece should just rest on the tape when the solenoid is not energized. The top of the solenoid should be just below the tape guide channel.

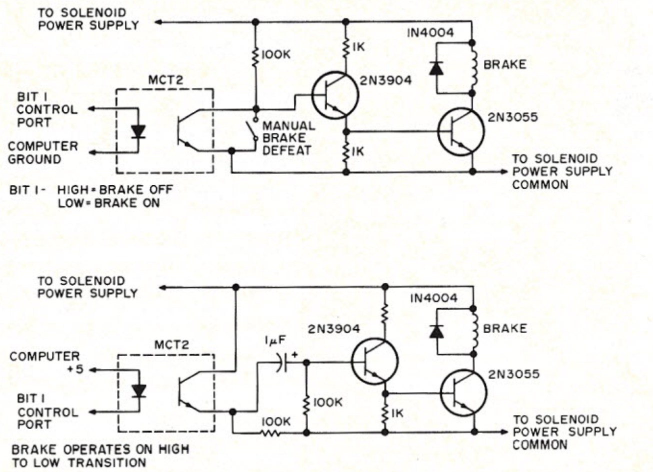


Fig. 9. Circuits to drive (a) continuous duty brakes and (b) pulsed brakes. The circuit in (b) generates a pulse about 0.1 sec long.

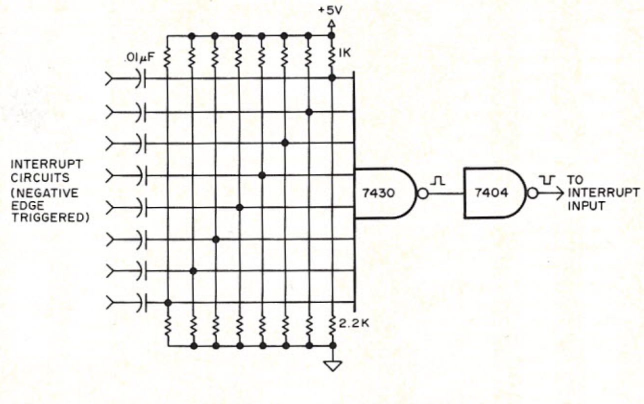


Fig. 10. Interrupt expansion circuit for an interrupt driven system. The output signal. (That is the leading edge for a ⊔ pulse and the trailing edge for a ⊓ pulse. The latter pulse must be 30 usec or longer to charge the RC network of the 0.01 capacitor and 2.2k resistor.) (From Phil Mork, "Interrupt Expansion Circuit" Micro-8 Newsletter, *Vol. 1, No. 4, p. 24.)*

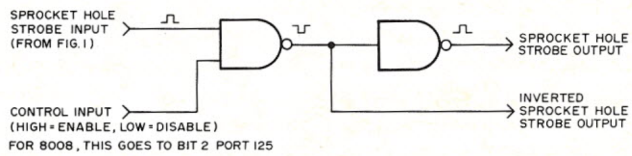


Fig. 11. Control schematic for the sprocket hole strobe pulses.

tapes using the Intel Hex format. The program is located at the last page of a 12K memory system. For other memory sizes, the page locations marked by asterisks (now 2F) should be changed to 0F for 4K. 1F for 8K etc. This program assumes the use of an I/O board with status bits similar to an Altair 88SIO Rev 0. For other boards, address 2F48 should be changed to reflect the position of the Data Available (DAV) bit and address 2F49 should reflect the polarity of the DAV bit (CA for positive DAV assertion and C2 for low DAV assertion). The DAV bit is set by the external input device to indicate the presence of valid input data. This bit is reset by the computer after accepting the input data.

Interfacing Without Interrupts

For an interface which does not use interrupts a different approach, which includes the control circuitry used in the interrupt driven method (Fig. 11), is used. If you have a spare I/O port the reader can be plugged into it with the sprocket hold output going to the input strobe connection. There are two more things to worry about. First, depending on how the computer resets the DAV bit, it may be left in an active state after reading a tape. If this is a problem, the same solution (Fig. 11) applies as for the interrupt interface.

This problem will only show up when the second tape is being read since the computer will immediately start reading in when the command is given. However, this is the same problem that you generate by putting the tape in and moving it on the leader. The only way this can get you into trouble is if your leader is not zeros (as in the Altair BASIC tapes). If this is a problem, the simplest fix is to position the tape so that a proper leader character is over the sensitive portion of

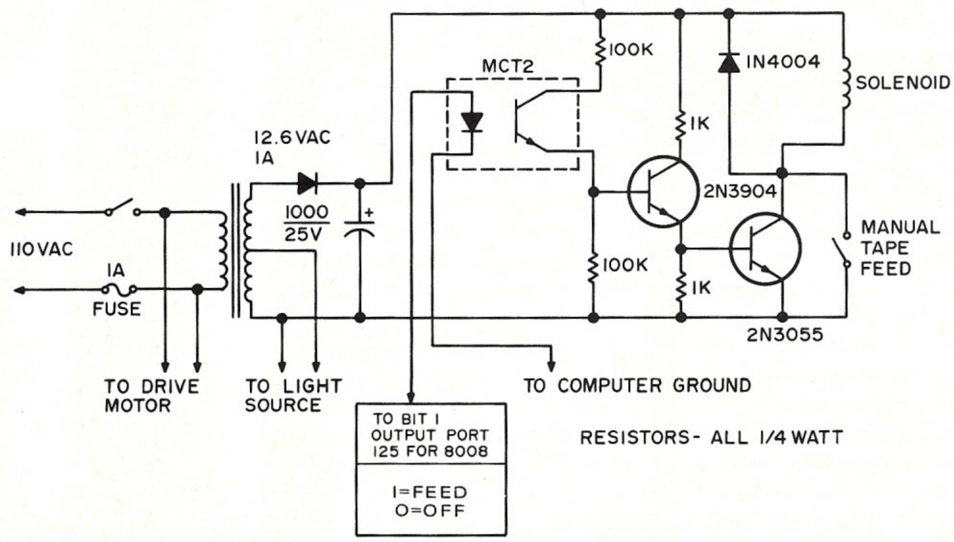


Fig. 12. Control circuit for the pinch roller solenoid. Note that the reader ground and the computer grounds are kept separate to eliminate electrical noise problems.

000	006	LAI	Turns
001	006	006	on paper
002	125	OUT CTRL	tape reader
003	000	HALT	
004	103	IN PTR	
005	074	CPI	Checks
006	000	000	to see
007	150	JTZ	if leader
010	003	003	
011	000	000	
012	000	HALT	Start High
013	103	IN PTR	address in H
014	350	LHA	
015	000	HALT	Start Low
016	103	IN PTR	address in L
017	360	LLA	
020	000	HALT	End high
021	103	IN PTR	address
022	330	LDA	in D
023	000	HALT	End low
024	103	IN PTRY	
024	103	IN PTR	
023	000	HALT	End low
024	000	IN PTR	address
025	340	LEA	in E
026	000	HALT	Load data
027	103	IN PTR	to memory

000	030	370	LMA	
	031	060	INL	
	032	110	JFZ	
	033	036	036	Increment
	034	000	000	address
	035	050	INH	
	036	306	LAL	
	03 7	300	NOOP	
	040	300	NO-OP	
	041	274	CPE	
	042	110	JFZ	
	043	026	026	
	044	000	000	Compare
	045	305	LAH	with end
	046	273	CPD	address
	047	110	JFZ	
	050	026	026	
	051	000	000	
	052	006	LAI	
	053	000	000	Turns off
	054	125	OUT CTRL	paper tape reader
	055	007	RET	Return to
				main program

Fig. 13. Input program for interrupt driven 8008 system with tape format described in the text. If not used as a subroutine, replace the RET statement with a HALT (000) and manually jump to the beginning address of the tape just loaded.

the phototransistors. This ensures the proper character will be read in whatever the previous state of the DAV bit.

The second problem is a little less obvious. You want the data input strobe to strobe the data into the computer on the edge of the strobe bit from the sprocket hole. If the DAV bit is set only by a level, not a transition, the computer is fast enough to read in many characters while the sprocket hole output is low. To correct this, insert a one shot (74121 not a 74123) in the sprocket hole output line and set it for a period of less than about 10 microseconds.

Alignment

If it is necessary to compensate for different sensitivities of the individual phototransistors in the package, use the following procedure: Place a piece of paper tape with all 8 holes punched over the sensitive portion of the phototransistors. With no pull up, resistors, R1-R8, adjust a light source above the array until the last LED just lights. At this light level, place a 100k pot between the gate input and the five volt supply. Adjust the pot until the LED has just turned on. Remove the pot, measure its value and replace with a fixed resistor of the same value. Those of you with lots of parts may want to just leave the pots in place. Make this adjustment for all of the data bits. The adjustment for the sprocket hole phototransistor is slightly different. It should turn on after the data bits so that when the data is strobed into the computer, the data bits are already stable. In this way, no latches or timing circuits are necessary. Adjust the pullup resistor R9 so that when the tape is moved through the reader the sprocket hole LED turns on after the data bits all come on but before any of them turn off.

If all this sounds like a lot of trouble, remember this procedure is necessary only if the diodes are not well matched and you feel compelled to use the almost transparent yellow tape. For black or gray tape you can generally set R1-R8 to 39K and set R9 to 33K.

After final assembly, the lamp-phototransistor distance must be adjusted. Put a piece of paper tape with all the holes punched over the sensitive parts of the phototransistor array. Adjust the lamp distance (or voltage) until all the LEDs have lit. Note this distance (voltage). Now put in a piece of blank light tape and adjust the distance (voltage) until any of the LEDs come on. Set the final distance (voltage) half way between these two points. This assures maximum noise immunity. If you are going to use only black tape do only the first measurement and then lower the lamp ¼" (increase the voltage 1 volt).

Summary

So that is all there is to building a paper tape reader. I can think of only one more caution. If you do not use the reader for an extended period (I know that this is unlikely but some people do get sick) be sure to remove any dust that may have collected on the surface of the phototransistor array. This is important because it is difficult to impress anyone when trying without success to load a tape for the fifth time.

The only hard-to-find component in this whole project is the phototransistor array. I have a supply of them and will be glad to sell them to anyone interested in building the unit at a price of $6.00 each. Send check or money order to: Doug Hogg, 36 Calle Capistrano, Santa Barbara CA 93105. ■

ADDRESS	DATA	COMMENTS
2F00	31 FF 2F*	LXI SP,2F00H
2F03	CD 06 2F*	CALL READ
2F06	CD 45 2F*	READ CALL TTYIN
2F09	FE 3A	CPI ':'
2F0B	C2 06 2F*	JNZ READ
2F0E	CD 2A 2F*	CALL CHAR
2F11	57	MOV D,A
2F12	C8	RZ
2F13	CD 2A 2F*	CALL CHAR
2F16	67	MOV H,A
2F17	CD 2A 2F*	CALL CHAR
2F1A	6F	MOV L,A
2F1B	CD 2A 2F*	CALL CHAR
2F1E	CD 2A 2F*	LOOP CALL CHAR
2F21	77	MOV M,A
2F22	23	INX H
2F23	15	DCR D
2F24	C2 1E 2F*	JNZ LOOP
2F27	C3 06 2F*	JMP READ
2F2A	CD 45 2F*	CHAR CALL TTYIN
2F2d	CD 3D 2F*	CALL HEX
2F30	07	RLC
2F31	17	RAL
2F32	17	RAL
2F33	17	RAL
2F34	5F	MOV E,A
2F35	CD 45 2F*	CALL TTYIN
2F38	CD 3D 2F*	CALL HEX
2F3B	83	ADD E
2F3C	C9	RET
2F3D	D6 30	HEX SUI 48
2F3F	FE 0A	CPI 10
2F41	D8	RC
2F42	D6 07	SUI 7
2F44	C9	RET
2F45	DB 00	TTYIN IN 0
2F47	E6 20	ANI 20H
2F49	CA 45 2F*	JZ TTYIN
2F4C	DB 01	IN 1
2F4E	D3 01	OUT 1
2F50	E6 7F	ANI 127
2F52	C9	RET

Fig. 14. Program for the 8080 to load Intel Hex Format paper tapes. The program assumes a control port at Port 0, with the Data Available bit (DAV) at bit 5. For other configurations, the instructions from 2F45 to 2F4F will have to be modified.

Why should a computer hobbyist subscribe to 73?

How does over 300 pages of articles on hobby computers published in 1976 grab you? ■ ■ ■ how many other magazines can claim that much? It may come as a big surprise to you, but there are a lot of non-ham readers of 73 ■ ■ ■ 73's barrage of computer articles has not entirely escaped the eyes of all computerists.

Yes, 73 certainly does cover the 25 or so hobbies which are classed together as amateur radio ■ ■ ■ radioteletype (RTTY), slow scan television (SSTV), DXing (contacting foreign countries), FAX (facsimile transmission), FM and repeaters, satellite use, moonbouncing, contests, experimenting, home construction, and even rag chewing. There are articles on building and using test equipment, on timers, weather satellite systems, wind speed measuring, and hundreds of other related and unrelated subjects.

73 is the largest of the ham magazines ■ ■ ■ the January 1977 issue ran 50 articles and was over 200 pages. Add to all of that the irreverence of Wayne Green and his editorials and you have a magazine that most readers read from cover to cover every month ■ ■ ■ no matter how many other magazines they get.

Do you really want to continue to miss all those computer articles?

SUBSCRIPTION TO 73 MAGAZINE

☐ $15 One year* ☐ $34 Three years* ☐ $150 Life

Name ______________________ Call (if any) __________

Address ______________________________________

City ____________________ State _______ Zip _______

*U.S. and Canada ONLY — write for foreign rates.

☐ Cash enclosed ☐ Check enclosed ☐ Money order enclosed

Charge to:

☐ Master Charge ☐ BankAmericard ☐ American Express

Credit Card # ____________________ Interbank # _______

Expiration date Signature ____________________________

☐ Bill me direct

YOU MAY USE THE CARD INSIDE THE BACK COVER FOR THIS ORDER.

73 MAGAZINE, PETERBOROUGH NH 03458 USA

Toll Free subscription number 800-258-5473 3/77

Computers for Free!

... schools have a better chance

There is simply no reason why every school in this country doesn't have a computer! Don points out some of the reasons why this isn't the case . . . but, more important, he also tells us how we can "go in the backdoor" (or any door) and remedy the situation. If you've got any kind of interest in making sure your kids get this education (and there can be little doubt of its value in the future) then read on. (Perhaps you could even let a few members of your local school board borrow your copy of Kilobaud *to read it, too!)*

By the way, Don is chairing the section on Personal Computers for Education at the First West Coast Computer Faire, and would like very much to hear from you if you have any contributions or a desire to participate. Call him at (408) 335-3360. – John.

Don Inman
350 Nelson Rd.
Scotts Valley CA 95066

Our public school systems today are under attack from all sides. Some argue that too much money is being spent. Some say that we should cut out all "frill" courses and stick to "basics." One of the objectives of a school system is to prepare students for the real world of work which they will encounter upon graduation. Admittedly most of us fail miserably at this. Changes are necessary. Yet any change in a school curriculum is looked at from a financial point of view. A new program is met with queries such as:

1. How much is it going to cost?
2. How much equipment is needed and how much will it cost?
3. Will teacher training be necessary and how much will it cost?
4. How many students can be accommodated and what will be the cost per pupil?
5. Will it require hiring additional teachers and how much will they cost?
6. Will the program force larger classes on other teachers?

etc.

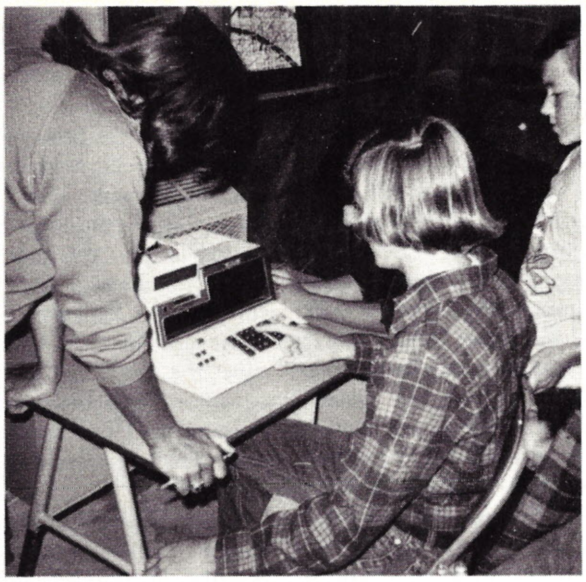
Seldom do we stop to ask, "What will the class do for the students?" Mention the words "computer class" and everyone concerned thinks of costs of thousands of dollars. They think of specialized teachers. Thoughts of new technologies trigger pangs of panic as old routines are forced out by new techniques. For these and other reasons, today's children are being kept on the fringes of one of the most exciting areas of learning.

The computer field is growing at a tremendous pace. Students of precollege age should possess some knowledge of what computers can and cannot do. They should have some idea about how they work. They should be developing a computer vocabulary. So many occupational fields are now making use of the computer that it is unthinkable that a student can graduate from our schools with a complete lack of knowledge in this area.

Computers are escaping from the arena of the specialized few. They are becoming commonplace in our daily lives. Yet, we continue to deny access to the computer to most school-age children.

Cost too much? Bah! Humbug! A school with one imaginative teacher can obtain computer equipment without spending a dime. He can equip the entire school system for less than the school spends for the fun and games of their athletic department, or band, or shops, etc. It has been done!

I know of several schools who have started with donated equipment and built



San Lorenzo Valley High School students investigating a SHARP Programmable calculator.

up a system that works. This article describes typical programs implemented with very limited budgets.

The first is a brief history of a development in my own high school. It is in its infancy but shows signs of developing into a successful venture. The most satisfying point of the story is that it all started with no money and some enterprising students.

Two years ago a group of twelve students in our school, San Lorenzo Valley High School, wanted to learn something about computer programming. When the powers that be were approached, the answer was, "Fine, but we have no money to spend on such a program." Students are not impressed with such answers. They were determined to have the course even if no equipment was available to them.

The instructor purchased some paperback textbooks on BASIC language which contained sketchy histories of computers, little on computer operation, but a good deal on BASIC programming.

We contacted several programming desk-top calculator companies to see what they had to offer. Representatives from Wang Labs, Hewlett Packard, and Monroe were surprisingly helpful even though they knew we had no money to spend. Equipment was loaned or demonstrated for time periods of a day to a week. The Wang 2200 was on display for a school open house. We tried everything from programmable calculators to a time-sharing computer system. What a wild time!

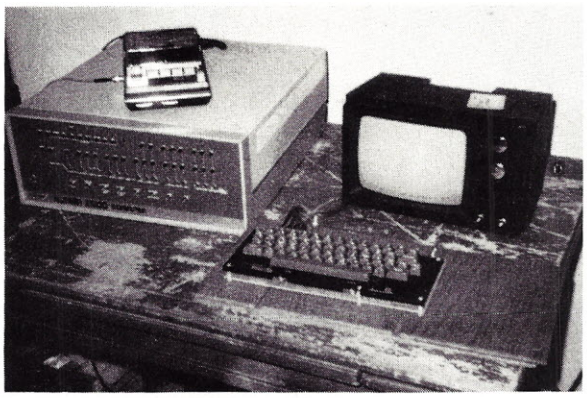
The course had no structure. The content was dictated by what vendor happened to be passing through town that week – no money being spent, but much learning going on, no matter how disorganized we were. But alas, we finally ran out of charitable vendors.

Teachers to the rescue! Three members of our math department decided the course was important enough to dig into their own pockets to finance a used programmable calculator from a local business equipment retailer. We formed an association and assessed ourselves enough dues to cover the down payment. Raffle tickets on some hand-held calculators brought in enough to make monthly payments for the balance of the school year. The entire math department budget for the next two years will take care of the balance.

We soon found that the programmable we bought was fine in a math classroom, but had little use elsewhere in the school. It did not have a high level language and was only keyboard programmable. We longed for a computer with BASIC, but we still had no money for it.

It was about this time that I discovered Bob Albrecht and the People's Computer Company (PCC). I wrote them a letter (which was promptly published) about our plight.

I must have struck a raw nerve as I described my ideas for a kit-building class for kids. Bob encouraged me, and I decided to find out what we could accomplish without spending any school money. Computer kits were appearing at that time and now seem to have flooded the marketplace.



Altair system owned by the Mountain Digital Group and used in the local school system.

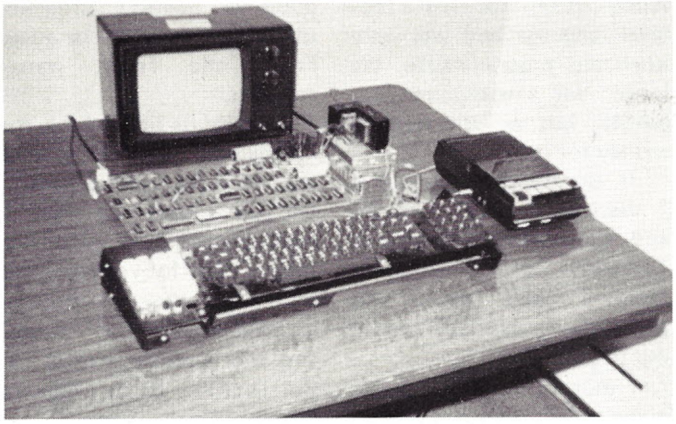
Our computer class was beginning to stagnate due to our lack of equipment. Two teachers and a parent formed what we call the Mountain Digital Group to try to stimulate interest in the community. We strongly feel that computer education should begin with the young people who lack the fear and prejudices that we have found in their elders. We were still unable to convince others to part with enough money to implement our ideas. Therefore (here we go again – I did say no money didn't I?) the three of us purchased an Altair 8800 by means of a kit-of-the-month plan. With each payment, we received one part of the computer. We finally ended the past school year with the Altair and 1K bytes of memory programmable in machine language from the front panel switches.

Despite its limitations, the kit-building idea for schools seemed to be striking a spark. Our equipment was used in high school classes, in an electronics class for youngsters at our local community college, in adult night school classes, and was demonstrated at the California State Science Teachers Convention.

To make the system practical for the school classroom, we have added a keyboard, CRT, and audio cassette. We are now buying 12K more of memory, a paper tape reader, and BASIC language capabilities. It has cost us some money, but the school district ended up with an operating system to use without expense.

Now how about that free computer system which I mentioned earlier? The class of computer students I had this past year decided they would start a project for the school. The object was to build a computer from kits, and we soon became known as the Kit-building Klass. It was at this point, that Bob Albrecht, the Head Dragon at PCC, came to our rescue. Through his efforts our school obtained a CPU board with all the trimmings from Microcomputer Associates, Inc. of Santa Clara.

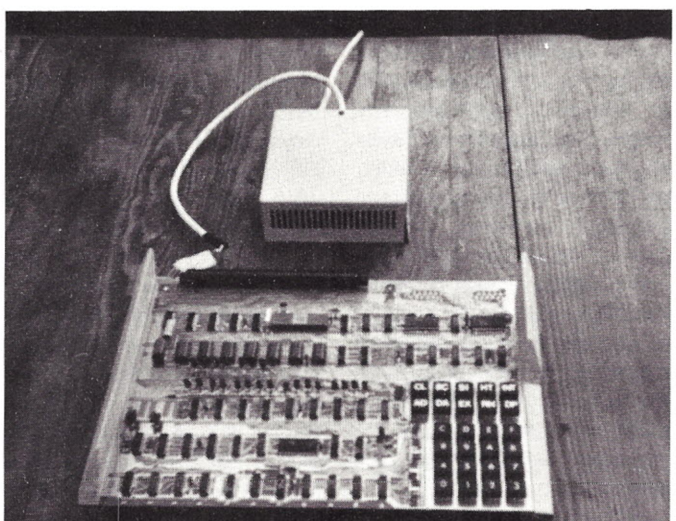
Four students volunteered for the project. They took turns laying out parts, stuffing the board, soldering,



Apple-1 computer with keyboard, videomonitor and cassette.

12-year old Kurt Inman (the author's son) contemplating construction of the Data Handler.

and inspecting each other's work. The students were all relatively new to electronics although three had limited experience in building simple kits. When the project was completed, they could all follow a schematic diagram in addition to a pictoral diagram. They used both to verify correct component placement on the printed circuit board. They could recognize and determine values of components. They became adept at soldering and even put together an article on their experiences which was published in People's Computer Company, May 1976.



The completed Data Handler.

The old saying that "It pays to advertise" certainly came true for us. Roger Rauskolb, a PCC reader and Hewlett-Packard employee, saw the story, phoned me, and donated a Jolt CPU board, along with 4K of RAM memory and 2K ROM memory containing a resident assembler. We're on our way. The Mountain Digital Group has furnished us with a power supply to run the system. Sometime during the year we hope to come up with some input/output device to get in and out of our Jolt computer. Our school will then have a computer of its very own, at no cost.

Who said it couldn't be done? Don't tell the kids that. They won't believe you. Our method was not orthodox. It did require that someone care enough to push the idea. Someone had to let the public know that our problem existed. In any new effort someone must stick his neck out even if he, or she, is a beginner in the particular field of endeavor. All three members of the Mountain Digital Group, which sponsors the computer efforts for our school, were new to computers a year ago. However they felt the introduction of computers to our students was important enough to spend their own money and time to get the ball rolling in our local school.

Publicity has been the key to acquiring aid from outside our local area. Our story finally made the local newspapers in July. We may now even find some local support for future projects. Sometimes you have to hit them over the head to get their attention. You have to keep pushing and not be turned aside when you receive no for an answer.

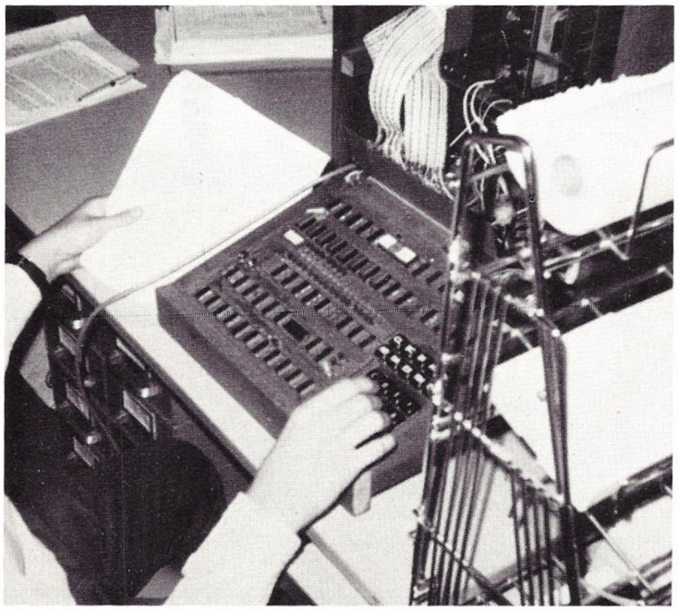
Our project has required much time and outside activity. There has been a lot of reading and seeking help from others. Nine times out of ten our requests for free equipment has been turned down, but that one yes is always worth the disappointments of the other nine. Our requests for free advice have never been refused.

With the rapid spread of microcomputers in kit form at a reduced cost, I see no reason any student should be denied access to some kind of computer. For those schools who are hesitant to use the kit-building approach, several systems can be purchased in assembled form. For example, have you seen the Apple-1? It is one of the neatest one-board, ready-to-go computer packages on the market. Plug in a keyboard and video monitor and you're ready to go. This computer is educationally affordable for most schools. Even after adding the keyboard, video monitor and cassette interface (and recorder), the cost should be under $1000.

The Apple-1 was developed by Steven Jobs and Stephan Wosniak, two young guys who wanted a computer of their own but couldn't afford to buy what was available. Working out of their garage, they built their own and then decided to market it. They have contributed to the ever growing list of newborn success stories in the computer field. The Apple Computer Company is

located in Palo Alto, California.

For the kit-builder there are many projects to choose from also. One that is within the price range of any school is the Data Handler by Western Data Systems. For $179.95 the Data Handler includes a large PC board with stand, 26 keyboard switches, complete set of ICs, a 6502 MOS Technology microprocessor, sockets for the memory chips, 1K bytes of memory, LEDs, resistors, capacitors, and a documentation package. A provision is provided to expand the basic one-board computer via a 100-pin connection to an Altair bus motherboard. Thus, existing Altair bus peripherals can be added.



The Data Handler being put to good use.

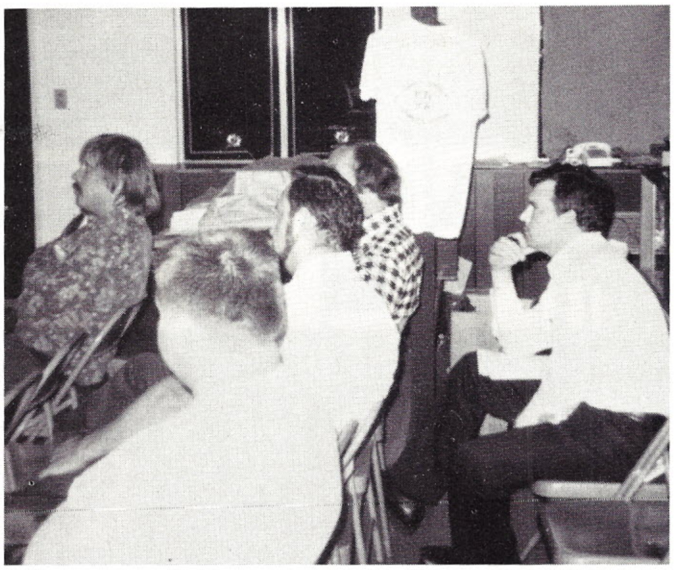
Western Data Systems is another group of young engineers, headed by Mike Indihar, located in Santa Clara.

One of the best resources for beginners and advanced computer buffs is the local retail computer store. These are springing up all over the country. They provide a rich source of aid and advice for those who feel they lack the knowledge to help students in this fast growing field. Many of the stores offer courses to compare available computers and also hold "how to" seminars. I recently sat through such a course at Byte Shop #2 in Santa Clara. The course was divided into four sessions covering:

1. **Getting Started** – a comparison of computers (cost, time to build, availability of software and peripherals, documentation available, and tools necessary for construction).
2. **Hardware** – with focus mainly on the 8080, 6800 and 6502 microprocessor chips.
3. **Software** – loading procedures, monitors, editors, and higher level languages.
4. **Interfacing** – connecting the microprocessor to the outside world through both serial and parallel methods.

Discussions were ably led by Todd Anderson, Byte Shop #2 owner, with provisions for frequent questions and answers. Individual problems encountered by the attendees were raised and possible solutions discussed.

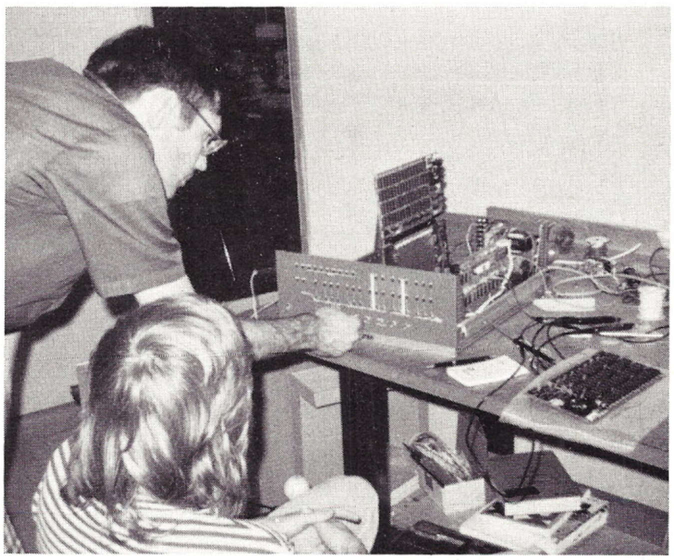
Another source of aid and advice is the hobby club in your area. Club newsletters present problems and solutions as well as helpful programs. The clubs offer a meeting ground for acquiring equipment, trading equipment, and for discussion of various microcomputer systems. Many clubs hold formal or informal opportunities to learn what computing is all about. Demonstrations of existing systems are frequently given.



Class at Byte Shop #2.

More and more computer related magazines and newspapers such as *Kilobaud, 73 Magazine, Creative Computing, PCC's* and *Dr. Dobb's Journal* are being published. Many of these have mail order book departments from which the latest in computer knowledge can be obtained.

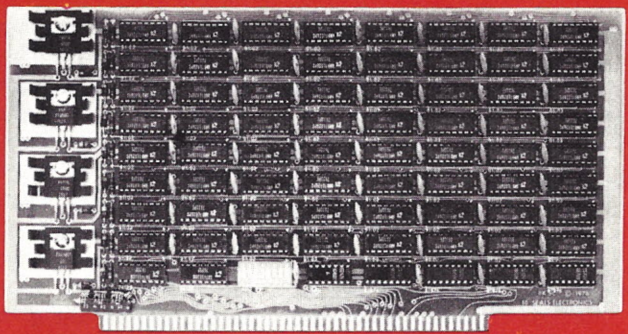
With the advance of low cost personal computers there seems to be no further excuse for holding back the education of our youngsters in this fascinating field. Build the kits in the science laboratory, teach programming in the math and business classes, and use the computers in every classroom where an application can be found. Let the students in on the act of developing their own equipment and course material. Here's the chance for teachers to be a real leader in innovative education. ■

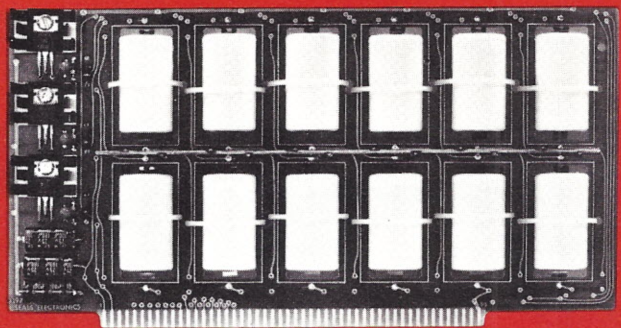


Todd Anderson, Byte Shop owner, checking Altair 8800.

SEALS ®

DOES IT





8 KSC (8 K STATIC MEMORY BOARD)

Our most popular item. Hundreds of satisfied customers. We have received an enormous number of letters praising our 8 KSC board. Our 8 KSC is undoubtedly the highest quality and most dependable board on the market today.

INTRODUCTORY OFFER

We are proud to announce that you can order your 8 KSC-Z with 250 ns memory at the same—yes SAME-price as our 500 ns memory.

		KIT	ASSEMBLED
8 KSC	500 ns	$295.00	$349.00
8 KSC-Z	250 ns	$295.00	$349.00
EXT extender card		$29.00	$38.00
100 pin edge conn. (Altair®)		$9.00	
100 pin edge conn. (IMSAI®)		$9.00	
Assembly & Operating Manual		$4.00	

BBUC (BATTERY BACK UP CARD)

ANOTHER FIRST

- Automatic battery charging circuit
- Selectable standby voltage outputs
- Will hold up to 12 "C" cell Ni-cad batteries. As much as 12 Amper hrs.
- The BBUC comes selected for 2.5 volts standby to pin # 14 on the S-100 buss structure, to power up the 8KSC memory
- Can be wired to back up any memory card which has battery standby capability. Even TWO polarities at one time
- Eliminate cluge wires on top of memory board. (Utilizes vacant buss lines)
- Just plug the BBUC into any available buss connector

★ HEAVY G-10 GLASS EPOXY PC BOARD
★ HEAVY PLATED THROUGH HOLES .5 mil. tin minimum
★ SOLDER MASK BOTH SIDES
★ COMPONENT LAYOUT SCREENED ON COMPONENT SIDE OF PC BOARD

KIT	ASSEMBLED
$55.00	$68.00

Assembly & Operating Manual $4.00

BATTERIES NOT INCLUDED

11 010 011 11 000 101 11 000 001 11 001100 11 010 011 · 11 010 011 11 000

TELEX # 55-7444 twx # 810/583-0075

Apple Introduces the First Low Cost Microcomputer System with a Video Terminal and 8K Bytes of RAM on a Single PC Card.

The Apple Computer. A truly complete microcomputer system on a single PC board. Based on the MOS Technology 6502 microprocessor, the Apple also has a built-in video terminal and sockets for 8K bytes of on-board RAM memory. With the addition of a keyboard and video monitor, you'll have an extremely powerful computer system that can be used for anything from developing programs to playing games or running BASIC.

Combining the computer, video terminal and dynamic memory on a single board has resulted in a large reduction in chip count, which means more reliability and lowered cost. Since the Apple comes fully assembled, tested & burned-in and has a complete power supply on-board, initial set-up is essentially "hassle free" and you can be running within minutes. At $666.66 (including 4K bytes RAM!) it opens many new possibilities for users and systems manufacturers.

You Don't Need an Expensive Teletype.

Using the built-in video terminal and keyboard interface, you avoid all the expense, noise and maintenance associated with a teletype. And the Apple video terminal is six times faster than a teletype, which means more throughput and less waiting. The Apple connects directly to a video monitor (or home TV with an inexpensive RF modulator) and displays 960 easy to read characters in 24 rows of 40 characters per line with automatic scrolling. The video display section contains its own 1K bytes of memory, so all the RAM memory is available for user programs. And the Keyboard Interface lets you use almost any ASCII-encoded keyboard.

The Apple Computer makes it possible for many people with limited budgets to step up to a video terminal as an I/O device for their computer.

No More Switches, No More Lights.

Compared to switches and LED's, a video terminal can display vast amounts of information simultaneously. The Apple video terminal can display the contents of 192 memory locations at once on the screen. And the firmware in PROMS enables you to enter, display and debug programs (all in hex) from the keyboard, rendering a front panel unnecessary. The firmware also allows your programs to print characters on the display, and since you'll be looking at letters and numbers instead of just LED's, the door is open to all kinds of alphanumeric software (i.e., Games and BASIC).

8K Bytes RAM in 16 Chips!

The Apple Computer uses the new 16-pin 4K dynamic memory chips. They are faster and take ¼ the space and power of even the low power 2102's (the memory chip that everyone else uses). That means 8K bytes in sixteen chips. It also means no more 28 amp power supplies.

The system is fully expandable to 65K via an edge connector which carries both the address and data busses, power supplies and all timing signals. All dynamic memory refreshing for both on and off-board memory is done automatically. Also, the Apple Computer can be upgraded to use the 16K chips when they become available. That's 32K bytes on-board RAM in 16 IC's—the equivalent of 256 2102's!

A Little Cassette Board That Works!

Unlike many other cassette boards on the marketplace, ours works every time. It plugs directly into the upright connector on the main board and stands only 2" tall. And since it is very fast (1500 bits per second), you can read or write 4K bytes in about 20 seconds. All timing is done in software, which results in crystal-controlled accuracy and uniformity from unit to unit.

Unlike some other cassette interfaces which require an expensive tape recorder, the Apple Cassette Interface works reliably with almost any audio-grade cassette recorder.

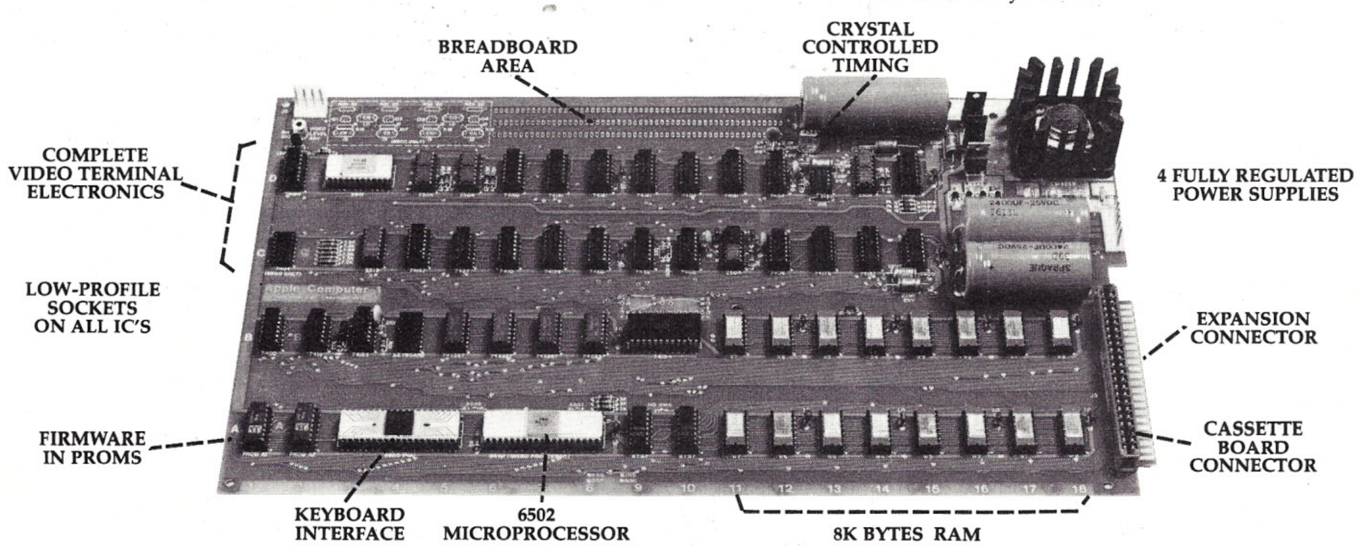
Software:

A tape of **APPLE BASIC** is included free with the Cassette Interface. Apple Basic features immediate error messages and fast execution, and lets you program in a higher level language immediately and without added cost. Also available **now** are a dis-assembler and many games, with many software packages, (including a macro assembler) in the works. And since our philosophy is to provide software for our machines free or at minimal cost, you won't be continually paying for access to this growing software library.

The Apple Computer is in stock at almost all major computer stores. (If your local computer store doesn't carry our products, encourage them or write us direct). **Dealer inquiries invited.**

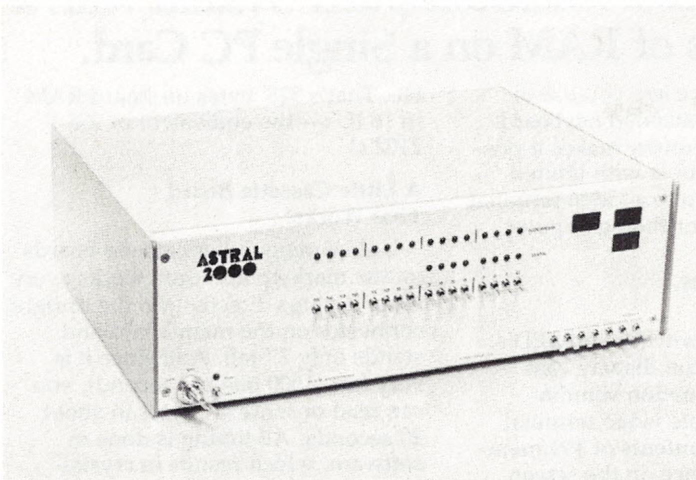
Byte into an Apple $666.66*

* includes 4K bytes RAM



APPLE Computer Company • 770 Welch Rd., Palo Alto, CA 94304 • (415) 326-4248

Sheila Clarke
P.O. Box 430
Glendale CA 91206



(Photo courtesy of M&R Enterprises.)

A New Approach to the 6800

... the Astral 2000

The following is an objective report on the Astral 2000 and M&R Enterprises. Kilobaud, *not M&R, asked Sheila to go up to San Francisco and write the article. — John.*

What ... *another* kit? Not exactly. The Astral 2000 advertisements announce that it's 70% assembled in kit form. Actually you can buy it mostly assembled for $995 or completely assembled for $1250. Which looks great for someone who's not too hardware oriented. In kit form the bulk of assembly is a nuts-and-bolts put together.

But *another computer?* Yup! Because M & R Enterprises, manufacturers of the Astral 2000, believes it is time to overcome what they feel are shortcomings of other computer systems involving physical configuration and function. They decided to implement long-time industry standards in designing this new system, creating versatility in software that would make the system useful for a wider range of applications than other microcomputers have been capable of to date.

So M & R Enterprises took the 6800 microprocessor, configured a design oriented toward industry standards and enlisted the programming services of a well reputed consultant. The price is highly competitive. Astral BASIC is well developed. And peripherals are available now. Although the system has been advertised for some months and several articles have appeared briefly describing the Astral, I've just heard that shipping has only begun as I write this in mid-November. Since Astral is just now appearing on the market, and as much as I'd have liked, I cannot produce testimony from Astral users. That makes it difficult to present an unbiased report. So without attempting judgement, you are asked to scrutinize what you read here, then write M & R Enterprises for more information at P.O. Box 61011, Sunnyvale CA 94088.

Design Philosophy

Marty Spergel of M & R Enterprises embarked upon his pursuit of "a better system" with very specific requirements. It had to coordinate well with standard hardware using industry accepted configurations. That is, it should fit easily into standard 19" racks and all cards used would be standard 4½" x 10" Vector Boards with 22 pin edge connectors.

Quality requirements, Marty feels, are the most stringent. Beginning with pretested ICs manufactured by Signetics under its "Superdip" program, all the way to final tests after burn-in, M & R feels the Astral 2000 is built to meet the most demanding high quality requirements for hardware assembly and reliability.

Spergel is proudest of Astral BASIC. He says, "... it is the most powerful ever

written that resides in 8K of memory." His requirements for the software included its ability to fit into 8K of memory and be more powerful than any existing BASIC residing even in 12K. It also had to be fairly fast and incorporate features that no other BASIC had within 8K or 12K. He wanted user-selectable precision in a floating point package, a powerful DO statement, trace mode capability, string facilities, comprehensive input/output and multidimension array capability.

The 6800 processor was chosen over the 8080 and 6502 because it was felt to work better, especially for word processing applications. When asked how he felt the Astral compared to other 6800 based systems, Spergel admitted that his opinions were based on what he'd read and heard because he'd never used other systems such as the Sphere or Southwest Technical Products. Although he feels the SwTP computer is good for developmental use, the Astral software doesn't tie up as much memory as the MIKBUG™, used in SWTP's computer.

Hardware Design

The basic system, whether a kit or assembled, includes the cabinet, motherboard, card rack, power supply, front panel board, processor board and an 8K RAM board. Components are individually wrapped in foil for shipment to prevent static damage.

The overall system is slightly smaller than many other systems, measuring 17" wide by 5" high by 12" deep. All cards used in the system are standard 4½", including the 8K RAM, CPU, PROM and Vid-80 video terminal. For the home brew hobbyist, this means easy acquisition of additional boards from most electronics and computer stores. Peripherals of course will utilize the same size boards.

The front panel combines the features of just about every other computer available, and adds a couple of its own. A set of 16-bit addressing switches leads directly to PIA ports. Switches are included for "Reset", "interrupt", "single step", "execute", "examine next", "load data", "load address", and "run/halt." An on/off locking switch prevents heart breaking goofs that often occur when a child comes too close. A 7-segment LED display operates under software control. An adjustable real-time clock is controlled through the PIA ports and adds features under program control to sequence processor actions for other equipment. This is useful for remote data acquisition, monitoring, and time controlled circuitry. It could be used, for instance, to control lighting and sprinkler systems while you're away.

An absolute clock is also built into the system which is divided by gates and is selectable in increments of seconds down to 10ths, 100ths, 1000ths, 10,000ths and millionths of a second. The user may install additional ICs for whatever time base he requires for his application.

The front panel snaps into the backplane connector which is located directly behind and parallel to the front panel. You'll do little soldering because there's no wiring, except to connect the power supply and bring power into the system.

The regulated power supply outputs 5 volts at 12 Amps and has its own overvoltage protection. Therefore, no onboard regulators are required in this system. Since it takes either 110 volts or 220 volts to run at 50 or 60 cycles, it can be used for overseas as well as domestic applications. The power supply also comes completely assembled, as already mentioned.

The processor board contains a serial I/O socket which outputs both RS-232 and 20 mA current loop, as well as signals for reader control and clock, and power source for peripherals. Power for system expansion has been provided via the entire bus structure for peripheral control, and memory operation.

The Astral employs a *cycle stealing* technique for direct memory access which is incorporated into the electronics of the processor board. Cycle stealing is a device used for applications which require high speed operation. The technique stretches the Ø1 and Ø2 processor clocks to provide a slowdown during which data transfer is accomplished. Without going into detail, this process allows the operator unlimited use of DMA transfers.

An interesting feature, though small, will certainly save hobbyists aggravating troubleshooting time. The system doesn't require a cooling fan because it is convection cooled.

The RAM board, which also comes fully assembled and tested, uses low power static RAMs. To be sure it'll work when you get it, the RAM board is put into a timing loop before it's put in the burn-in rack. After 48 to 72 hours of burn-in time, it is tested again. Since failures will doubtlessly show up after such rigorous scrutiny, it's certain to work when the buyer receives it.

Assembly

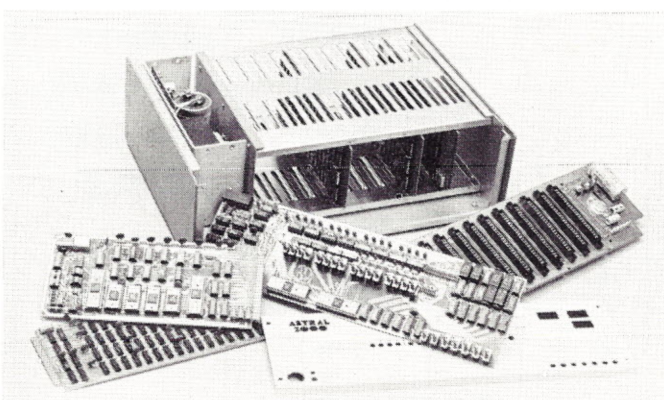
They say it takes just 15 minutes to assemble the cabinet, chassis and card cage. You'll need a little electronics know-how to bolt in the power supply and wire about ten components to the motherboard, then wire in the power supply to the front panel. Other than that, all that's necessary is loading about 52 ICs into their sockets. Soldering should create no problems since all the boards are solder masked.

Although I haven't seen the documentation, I'm assured that the instruction manual will include information such as the best solder iron and type of solder to use. It's promised to be complete with photos and illustrations including a picture of the finished product.

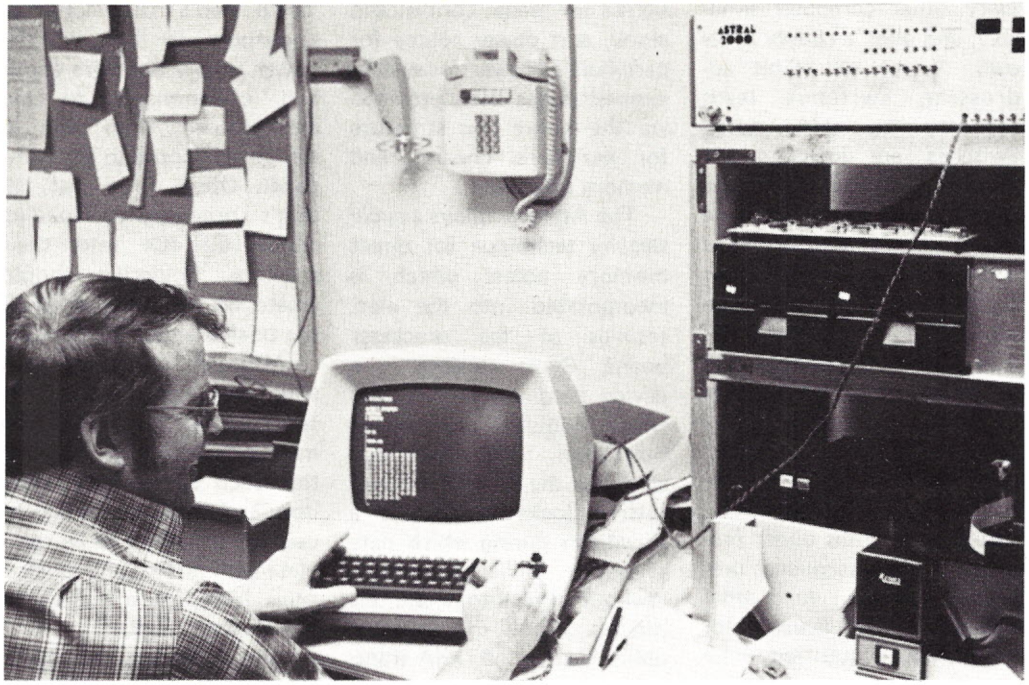
Also included will be a step-by-step diagram of the motherboard and front panel board assemblies with orientation of the ICs. Marty Spergel doesn't promise what he's called, "Heathkit overkill," but he feels the manual is sufficiently complete for experienced hobbyists.

Memory

The Astral will hold as many as 12 cards, which should leave sufficient room for maximum memory capacity. The system will take 64K bytes of memory, 56K bytes of which is programmable and 8K of which is taken up with Astral BASIC. Additional 8K RAM boards are available fully assembled at $245 each. 8K EPROM boards are also available at $59.95 each. Don't let that price throw you . . . the memory chips aren't in-



(Photo courtesy of M&R Enterprises.)



The complete system uses an ADM-3 terminal. To make it easier for this user, he has rewired the reset switch for his convenience.

cluded, though sockets for them are. The EPROM boards use 5204 electrically programmable ROMs. (See Fig. 1)

Peripherals

All peripherals offered for the Astral are assembled and tested and plug directly into the system. Now available is the VID-80, a video terminal available as an option for $189.95. It has selectable line lengths of 64, 72 and 80 characters per line. It displays 16 lines of upper case characters with provision for optional lower case character ROM. It will also utilize a light pen for CRT graphics and word processing.

Fig. 1.
The Astral 2000 memory map accommodates up to seven 8K memory boards. Both RAM and EPROM boards may be used and will handle peripheral I/O devices and several more capabilities.

NUMBER OF BYTES	ADDRESS RANGE	MEMORY AREA	FUNCTION
256	FFFF – FF00	FRONT PANEL PROM	FRONT PANEL & CLOCK FIRMWARE
8	FE13, FE10, FE0B, FE08	FRONT PANEL / PIA'S	FRONT PANEL I/O PORTS, ETC.
4	FD13 – FD10	RESERVED FOR VID-80	VIDEO DISPLAY BOARD INPUT PORTS
1.3K	FCFF – F800	RESERVED FOR FUTURE SYSTEM EXPANSION	RESERVED FOR FLOPPY DISK CONTROLLER, USER I/O PORTS, ETC.
2K	F7FF – F000	RESERVED FOR USER I/O	RESERVED FOR FLOPPY DISK CONTROLLER ROM
2K	EFFF – E800	MONITOR ROM	SYSTEM DEBUG AND MONITOR FIRMWARE
4	E403 – E400	MONITOR I/O PIA	MONITOR I/O PORT
256	E07F – E000	MONITOR STACK RAM	MONITOR STACK & TEMPORARY STORAGE
56K	DFFF, 00FF, 0000	RAM / DIRECT ACCESS MEMORY	RAM STORAGE

The cassette I/O interface, also available as an option for $49.95, plugs directly into the serial I/O socket on the processor board and permits use of most low cost cassette recorders.

Peripherals planned for the future include a floppy disk with a bus compatible controller card. A single drive unit will include power supply, cabinet and cable for about $795. M & R Enterprises expects to be able to offer a dual drive floppy about next spring for under $1,000. The Astral will handle up to three drives with the software provided.

A color graphics board is also planned, which is to have 16 intensities and more than 8 colors. A 16-channel AD/DA converter will work together with the VID-80 to move images from right to left and from left to right. It'll also utilize light pen capability. A typical application for this option might be a scientific one which uses animated charts and graphs.

Using The Astral 2000

System Monitor — A 2K x 8K monitor in ROM is shipped with the computer and can be interfaced with the Vid-80 terminal or the serial I/O port. Debugging facilities for machine language programming will be useful for the low level language programmer, and include patching and inserting code, capabilities for breakpoints and breakpoint stepping, and provisions for displaying the contents of memory.

Other aspects of the Astral monitor include:

Command Intercept — lets user add his own commands.

Character Intercept — lets user extend video display software for editing.

Check Sum — may be calculated for entire block of memory.

Block Insert — permits temporary insertion of a set of instructions for experimentation.

Memory Alter — automatically calculates branch offsets.

Also included are provisions for intercepting software interrupt, nonmaskable interrupt and maskable interrupt.

A feature M & R Enterprises is proud of was built into the monitor for keystroke efficiency. It permits commands to be repeated by entering a "TYPE 8" command and pressing the space bar. This allows the user to examine successive 8-byte chunks of data rather than

having to re-enter each location for the next 8 bytes.

Astral BASIC – The entire Astral software package may be purchased with the initial system for $35 and BASIC will arrive either on cassette tape or paper tape, whichever is requested. Purchasing the software entitles the buyer to a year's subscription to the Astral Newsletter which keeps owners abreast of updates and new programs.

Astral BASIC resides in 8K of RAM, thus making it necessary, as we've already mentioned, to buy an optional 8K RAM board when buying the initial system in order to have programmable memory.

This extended BASIC was written especially for the Astral 2000 system and looks to be fairly powerful. It's versatile enough to satisfy application needs ranging from game playing to involved business systems. A few of its features include:

Floating Point Package – User selectable precision lets the programmer select from 6, 9, 13 or 16 digits. Using 16 digits enables the programmer to compute figures exceeding the gross national product, but is slower than using a lower number. Using 9 or 13 digits would enable one to program for detailed business applications at a fairly high rate of speed. Interestingly, most BASICs use only 6 digits. Increasing the number of digits does not require an increase of memory.

DO Statement – Lets the user subroutine without using separate subroutines. Although subroutine features are also found in other versions of BASIC which are included in Astral BASIC, this one's different. Its format would be: DO label-1, label-2. After execution, control is returned to the terminal.

Trace Mode – Useful for debugging. List statement line numbers are listed as executed. They may be programmed to TRACE ON for subroutines that still require checkout, and TRACE OFF for routines that have been vertified. Pressing the escape key returns control to the terminal.

Variable Length Strings – and capability of searching for substrings are useful for word processing applications.

Edit – permits loading, listing and saving of programs. Delete or renumber blocks of statements, such as insertion of a line between two consecutive lines, which will wind up renumbered in numerical order.

Print Using – is useful for business applications in that it allows floating dollar signs, plus and minus signs, and floating commas for special formats. This lets the user print numbers in standard accounting format, i.e., $1,000,000+.

Conclusion

So maybe the Astral 2000 is not *just* another computer. It might come closer to being a turnkey system for the hobbyist than has yet been available, because the software provides a greater range of versatility than we've seen in a single system so far. Forgive me if that sounds like a judgement on my part, but that is how I was impressed when I saw the system working. If I was disappointed at the time (summer's end) that the Astral wasn't being shipped yet, nor were orders being taken, I was impressed that the manufacturer wanted to hold out until everything was exactly right for shipments of guaranteed systems.

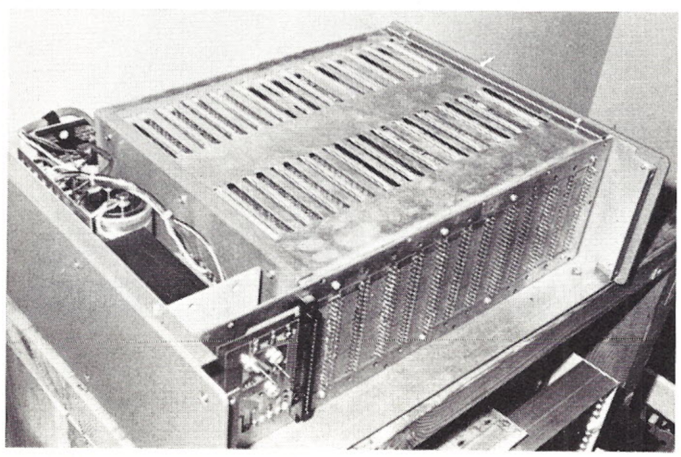
One policy M & R Enterprises holds which hobbyists may not agree with is the way in which they are marketing the Astral 2000. Dealers will not be handling it. M & R will take orders directly with promised delivery from stock to 8 weeks. As far as I'm concerned, that's a back-slide in customer service technique. It invites customer grumblings of being out $1,000 while awaiting the paid-for computer.

I don't know how that policy will affect service. As a hobbyist, it would seem much easier to walk into the computer store, where the system might have been purchased, to troubleshoot hardware and software problems. Dealers must be serving that purpose; new ones are still opening all over the country. The only reason, I deduce, M & R is bypassing the middleman might be to save the markup and hopefully, pass that savings on to the buyer.

If you want some form of personal recommendation or criticism of the system, you might contact Marty Spergel's hobby base of operation, the Homebrew Computer Club in Mountain View, California. Many of its members are familiar with the system and what went into its development. Some may be Astral owners by now.

With no sales and use history to reflect upon, I'll leave it to you pioneering enthusiasts to determine for yourselves whether the Astral 2000 is for you. ■

Prior to shipment of orders, here is one of the few Astral 2000s in operation. Its I/O devices here include a dual drive floppy and a paper tape reader.



Removal of the cover reveals the Astral 2000's backplane and power assembly.

This is probably one of the best articles you'll run across in quite a while if you're one of those with a lot of questions about the internal workings of a microprocessor. Some of the analogies Lance uses in explaining certain concepts and hardware are just great! If you've already got the background covered here, be sure and refer newcomers to this article when they ask you about good introductory material. — John.

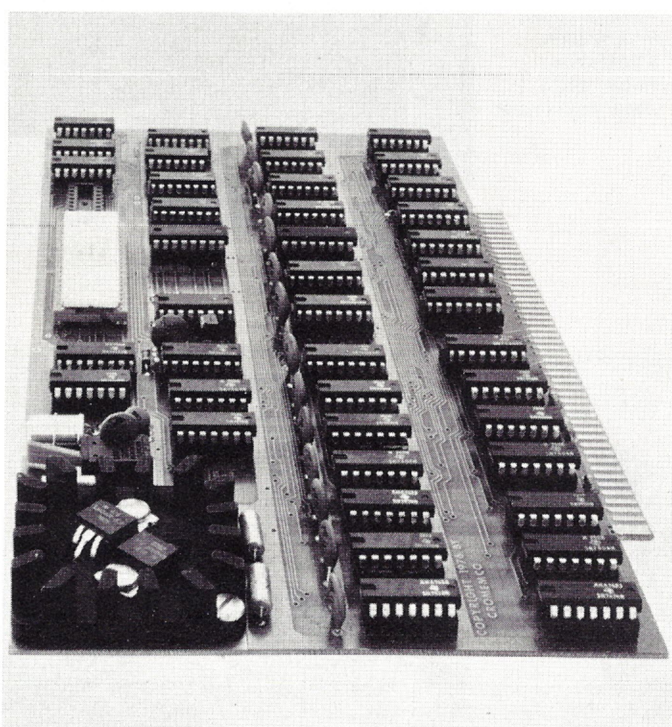
Journey into the CPU

... the view from within

Dr. Lance Leventhal
138 South Acacia Ave.
Solana Beach CA 92075

The previous article* in this series described the memory and input/output sections of a computer. This article discusses the control section and the registers that define its architecture. We will then describe the arithmetic circuitry and will conclude by showing how the CPU decodes and executes instructions.

Although the CPU represents a small part of the size and cost of most computer systems, it provides the *brains* for the entire computer and its capabilities determine the capabilities of the system. A new CPU card (like the Zilog Z-80 card from Cromemco) can upgrade a computer system like the Mits Altair 8800B. The CPU determines the speed of the computer, its I/O and memory capacity, and the ease with which it can be programmed and interfaced.



The Cromemco Z-80 CPU Board. Courtesy of Cromemco Inc., Mountain View, CA.

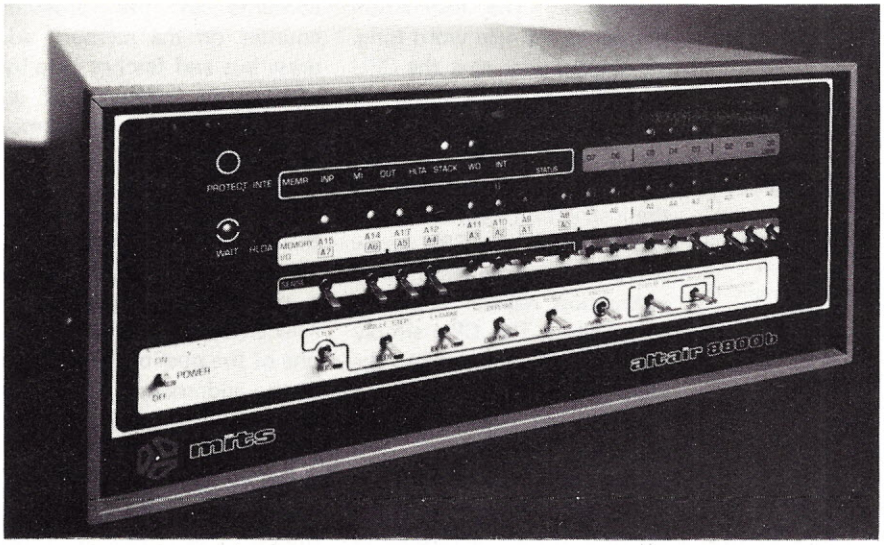
*"Well, Your Micro's Built ... where do you grow from here," *Kilobaud*, January 1977.

Functions of the CPU

The control section of a computer has a large number of functions. It must:

1. Fetch and decode instructions.
2. Transfer data to and from the memory and input/output sections.
3. Perform arithmetic and logical operations.
4. Determine internal conditions (flags) resulting from the operations of the CPU.
5. Recognize external control signals such as interrupts and startup (or reset).
6. Provide timing and control signals.

The CPU must contain the circuitry required for all these tasks. The primary sections of the CPU are the register structure which serves to hold data for internal use, the instruction decoding mechanism which converts the instructions into the required internal and external control signals, and the arithmetic-logic unit (ALU) which performs the arithmetic and logical



The MITS Altair 8800B, a popular microcomputer based on the Intel 8080 CPU. Courtesy of MITS Inc., Albuquerque, NM.

operations. We will discuss the register structure and then the arithmetic unit and the instruction decoding mechanism. Fig. 1 shows the components of a computer control section and their connections.

Register Structure

The register structure of a computer is a key element in its architecture. Registers are small memories which are inside the control section. They hold results which are used or generated at certain times during the CPU's cycle and must be saved during the rest of the cycle. Registers can also store frequently used data or addresses; the CPU can then use those quantities in its operations without performing a memory access. Registers are connected to each other, to the ALU, and to the external buses by means of internal buses. Registers are controlled by clock and enable signals. The contents of a register are unaffected by input data when the clock signal is inactive. When the clock signal is active, the input data is transferred into the register and latched (or held) by the transition of the clock to the inactive level. The clock signal thus serves to open and close the gate which allows data to enter the register. Clocking registers is a common function of timing and control signals.

Register outputs are usually buffered. Here, a control (or enable) signal determines whether data is transferred from the register through the buffer. The enable signal opens and closes the gate which allows data to leave the register. Buffer control is another common function of the CPU's internal timing and control signals. Fig. 2 shows the input and output control of a register.

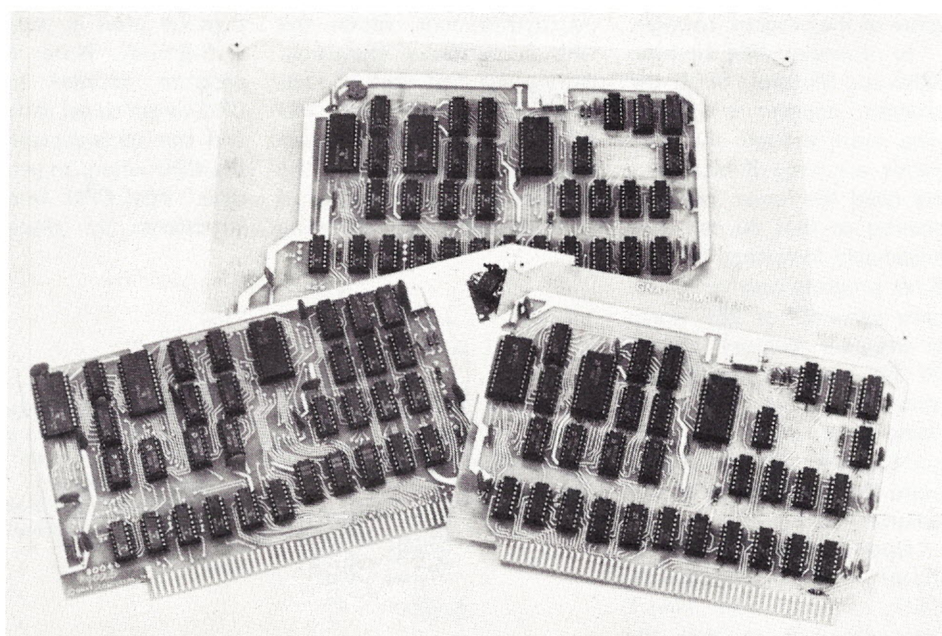
Types of Registers

Most computers have many different types of registers; the most common are:

1. Program counters
2. Instruction registers
3. Address registers
4. Accumulators
5. General-purpose registers
6. Index registers
7. Stack pointers
8. Condition code registers

Even the simplest microprocessor typically has several types of registers.

The *program counter* (abbreviated PC) contains the address of the next word of program memory which the CPU will fetch. Each CPU instruction cycle starts with the program counter being placed on the address bus so that the CPU can fetch the first word of the instruction. The CPU automatically increments the program counter



A hardware multiply/divide module for Altair or IMSAI computers. The module, based on TTL logic, performs multiplication or division in just 5 microseconds (as compared to over 200 microseconds in normal software). It costs $225. Courtesy of Gnat Computers, San Diego, CA.

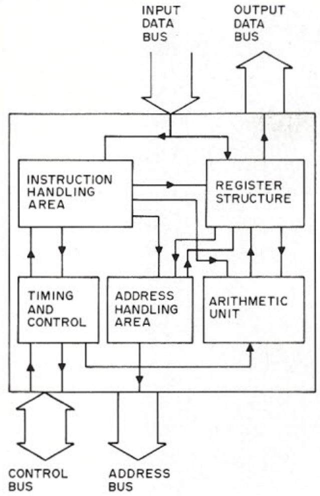


Fig. 1. A typical CPU with its various components and connections.

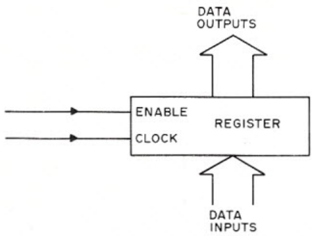


Fig. 2. Register input and output control. The register normally serves as a small memory. The CLOCK signal controls the flow of data into the register; ENABLE signal controls flow of data from register. Internal timing circuitry and the instruction decoding mechanism produce the CLOCK and ENABLE signals.

after each use; thus the CPU steps through the program sequentially unless an instruction specifically alters the value of the program counter.

In processors like the Intel 8080 and Motorola 6800, the program counter is 16 bits long even though the data words are only 8 bits long. We need the longer program counter so that we can have reasonably long programs. An 8 bit program counter would only allow 2^8 or 256 words or program memory while a 16 bit program counter allows 2^{16} or 64K. 256 words could only handle tiny programs while 64K is usually more than enough for all our program needs.

Note that the function of JUMP, CALL, BRANCH, SKIP, WAIT, and HALT instructions is to stop the sequential execution of programs. The computer will proceed relentlessly (like a robot in a science fiction movie) unless we have a way to stop it. It will keep on executing instructions sequentially, passing completely through the memory and starting all over again.

JUMP, CALL, BRANCH, and SKIP instructions place new values in the program counter so that the computer will repeat sections of the program or perform instructions out of their normal order. WAIT and HALT instructions cause the computer to stop incrementing the program counter and just *stand in place* until external signal orders it to proceed.

The *instruction register* (abbreviated IR) contains the instruction code which the CPU is currently executing. During the first part of each instruction cycle, the CPU fetches a word from program memory and places it in the instruction register. The word is then sent to the decoding mechanism. The instruction register is a single word long. We should note that the CPU uses the instruction register automatically during each cycle. The programmer seldom has or wants access to the instruction register.

How does the CPU know whether to place data in an instruction register or somewhere else? The CPU simply has a switch connected to the input data bus which directs the data to one of several possible destinations. The switch position is determined by select inputs from the internal timing and control signals. We call this switch a *demultiplexer* since it takes data from a shared bus and sends the data to the proper destination. The demultiplexer has no more intelligence than a railroad switch. It directs the data without examining it in any way. Fig. 3 shows a simple 2-way demultiplexer which sends data either to the instruction register or to the arithmetic unit.

Address registers (usually abbreviated MAR for memory address register) hold the addresses of data which is being fetched from or sent to memory. The CPU may place the address portion of the instruction into an address register or the programmer may place a value in the address register which will then be used by subsequent instructions. Note that the program counter tells the CPU where to get instructions and the address register tells the CPU where to get or send data. The CPU fetches instructions by placing the contents of the program counter on the memory address bus and fetches data by placing the contents of an address register on the memory address bus.

The CPU uses a switch connected to the address bus to determine which source it will use for the address during a particular memory cycle. This switch directs data from one of the possible sources on to the address bus. It has the opposite function of the switch mentioned earlier since it gets data from a source and places it on a shared bus. We call this switch a *multiplexer* or *selector*. It chooses an input on the basis of internal timing and control signals. Fig. 4 shows a simple multiplexer with two sources; this multiplexer can get an address either from the program counter or from an address register.

An *accumulator* is much like the register which holds the total in a calculator. The accumulator serves as a source and destination for most computer instructions. When the CPU performs an addition instruction, it adds the contents of the specified memory or register address to the contents of an accumulator and places the result back in the accumulator. The operation is similar to the hand calculator where we add a new entry to an old subtotal to get a new total. Accumulators serve many other purposes in most small computers; they often are a source or destination for data transfers, shifts, and special-purpose instructions. Accumulators are the center of the processing activity of the CPU.

A *general-purpose register* is a register which can serve many functions under the control of the programmer. We use general-purpose registers to store data which we use frequently and to hold partial results after we have calculated them in the accumulator. Some computers allow the programmer to assign

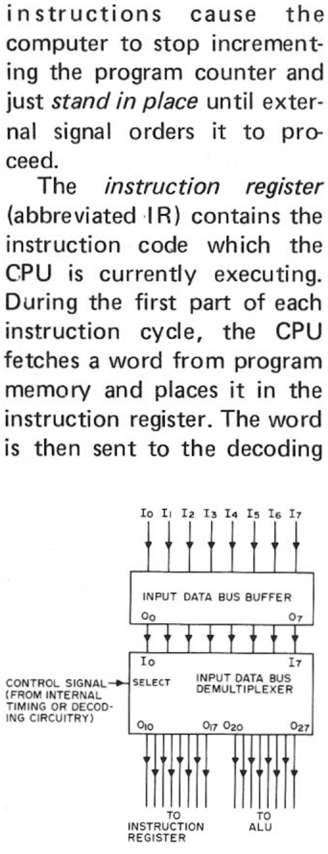


Fig. 3. A simple demultiplexer. If the select input is zero, the data is sent to the instruction register; if the select input is one, the data is sent to the ALU.

functions to a general-purpose register; the programmer may then declare that a particular register is an accumulator, address register, index register, or even program counter. These assignments let the programmer configure the computer flexibly to meet particular program needs; the DEC PDP-11, RCA COSMAC, and General Instrument CP1600 are examples of minicomputers and microprocessors that work this way. However, we will obviously have an easier time using and understanding a computer that has fixed register assignments.

An *index register* is a register which is used to identify particular elements in an array or table. We may want to handle all or any of the elements of such data structures with the same instructions. We do this by placing the base address of the array in the instruction and the index of a particular element in the index register. The computer adds the base address to the index register to get the actual address of a particular element.

The process is similar to the common mathematical technique of referring to an element of an array as a subscripted variable, e.g. as A_i where A identifies the array and i the particular element. The two parts of the address are the answers to two separate questions, i.e. which array or table, and which element in that array or table? The problem is like finding an entry in an encyclopedia by specifying first the volume number and then the page number within the volume. Note that we can process a different element with the same instruction by changing the index without changing anything else.

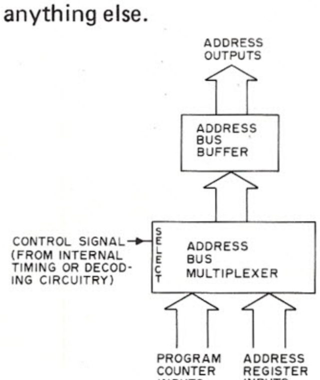


Fig. 4. A simple multiplexer. If the select input is zero, the multiplexer places the contents of the program counter on the address bus; if the select input is one, the multiplexer places the contents of the address register on the address bus.

A *stack pointer* is a register which contains the address of the top element in a last-in, first-out stack. The stack operates like the spring-loaded plate holder in a cafeteria or like a paper cup dispenser. Even these simple examples show that a stack can operate either upward or downward. When we add an element to the stack, we put it in the next sequential address in memory and adjust the stack pointer accordingly (we may adjust the stack pointer up or down depending on which way the stack grows). When we remove an element from the stack, we get it from the address in the stack pointer and adjust the stack pointer so that it now points to the next element in the stack.

The advantages of the stack are that we can place data in it without destroying the old contents and we can remove data from it in the opposite order from which we entered it. We must be careful, however, to remember both the order and the identity of the elements we enter in the stack. Note that the CPU can place markers in the stack and retrace its steps, since the markers will come out in the opposite order from that in which they were entered.

A *condition code register* is a register which contains single bits or *flags* which tell whether or not certain conditions exist within the computer. Common flags are:

CARRY – 1 if the last arithmetic operation generated a carry from the most significant bit. We use the CARRY to perform multiple-precision arithmetic and to examine or transfer single bits of information.

HALF-CARRY – 1 if the last arithmetic operation generated a carry from bit 3. We use the HALF-CARRY only on 8 bit CPUs like the Intel 8080 or Motorola 6800 to perform decimal arithmetic.

INTERRUPT ENABLE – 1 if an interrupt will be accepted. We use the INTERRUPT ENABLE to keep interrupts out when they would interfere with other system functions.

OVERFLOW – 1 if the last arithmetic operation produced a twos complement overflow. We use the OVER-

FLOW to perform signed arithmetic.

PARITY – 1 if the last operation produced a result with an even (if even parity) or odd (if odd parity) number of 1 bits. We use PARITY in character manipulation and communications to provide a check on possible transmission errors.

SIGN (or NEGATIVE) – 1 if the last operation produced a result with a most significant bit (sign bit) of 1. We use the SIGN to perform signed arithmetic and to examine single bit of information.

ZERO – 1 if the last operation produced a zero result. We use the ZERO bit to control loops and perform comparisons. (Note that the ZERO bit is *1* if the result was *zero*).

Some computers have other status flags as well. However, we use the CARRY and ZERO most frequently; the SIGN, OVERFLOW, and INTERRUPT ENABLE occasionally and the others hardly at all.

The most common use of flags is to select sequences of instructions to be executed depending on the flag values. A conditional JUMP instruction (e.g., JUMP ON FLAG TRUE) causes the CPU to execute one set of instructions or another by placing a new value in the program counter only if the flag has the specified value. Typical conditional JUMP instructions are JUMP ON ZERO, JUMP ON NOT ZERO, JUMP ON CARRY, and JUMP ON NOT CARRY. These instructions form the basic decision-making capability of the computer.

Examples of Register Sets

Particular CPUs will have a set of registers including many of the types we have described. The Intel 8080, for example, has (see Fig. 5) 1 16-bit program counter, 1 8-bit instruction register, 1 8-bit accumulator, 6 8-bit general-purpose registers which can also be used as 3 16-bit address registers, 1 16-bit stack pointer, and 5 status flags: CARRY, ZERO, PARITY, SIGN, and HALF-CARRY.

The Motorola 6800 has (see Fig. 6) 1 16-bit program counter, 1 8-bit instruction register, 2 8-bit accumulators, 1 16-bit index register, 1 16-bit stack pointer, and 6 status flags: CARRY, ZERO, OVERFLOW, SIGN (NEGATIVE), HALF-CARRY, and INTERRUPT.

Note the tremendous similarities between these rather different processors. Small variations in the register structure can have an enormous effect on the ultimate performance and characteristics of a CPU.

Arithmetic Area

The arithmetic area of the CPU actually performs the arithmetic, logical, shift, comparison, and special-purpose instructions. The arithmetic area is where most of the data manipulation (or actual processing) takes place. The instructions are decoded to provide the control signals required to select the operands and operations for the arithmetic area.

Traditionally, the arithmetic area of a computer consisted of a large amount of hardware which could only perform a few simple functions directly. The basic components of such an arithmetic area were:

1. A binary adder which could add two binary numbers.

2. A logical complementer or inverter which could invert a binary number (i.e. replace all the zero bits with ones and the ones with zeroes).

3. A shifter which could shift a binary number one position to the right (a left shift can be performed by adding the number to itself).

This arithmetic circuitry can provide all the common data processing functions although most will require a series of internal cycles or instructions. The binary adder, of course, can directly perform binary addition; together with the complementer, it can also perform binary subtraction by passing the number to be subtracted through the complementer and then adding it to the other number.

Together with the shifter, the adder can also perform mulitiplications and divisions as a series of additions and subtractions respectively much as we perform these operations by hand. Our placing of results to the left or right to indicate more or less significant digits corresponds to the computer shifting data to the left or right. The shifter can also be used to move bits to positions where they can easily be processed, combined for storage, or used as output data.

The basic arithmetic unit can easily produce the most widely used flag or condition bits:

1. The CARRY bit is the carry from the addition of the most significant bits in the binary adder.

2. The ZERO bit is the logical OR of all the bit results from the binary adder or all the bits in the accumulator.

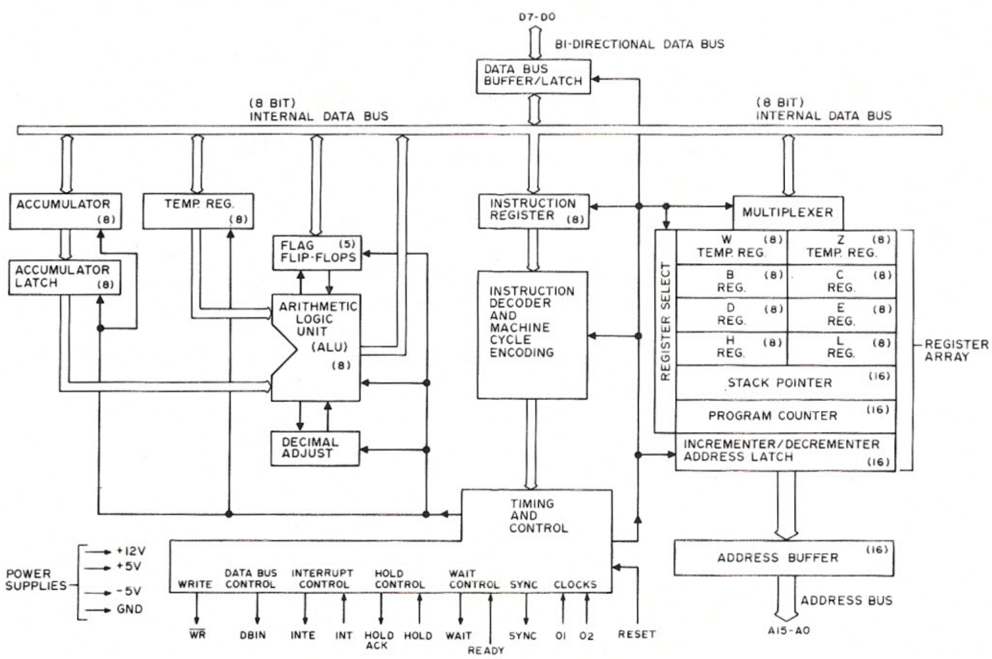


Fig. 5. Intel 8080 CPU Functional Block Diagram. Courtesy of Intel Corp., Santa Clara, CA.

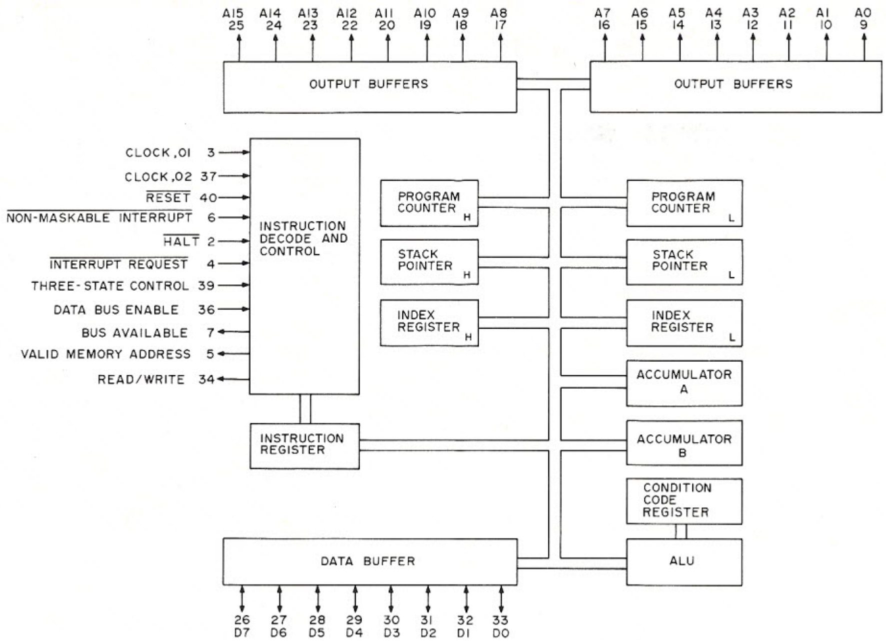


Fig. 6. Expanded Block Diagram of the Motorola 6800 CPU. Courtesy of Motorola Semiconductor Products Inc., Phoenix, Arizona.

The output of the OR gate is usually inverted to give a ZERO bit that is 1 if and only if all the bits in the adder result or the accumulator are zero.

3. The SIGN (or NEGATIVE) bit is the most significant (left-most) bit of the result from the binary adder or the most significant bit of the accumulator.

Obviously, all except the simplest instructions will execute rather slowly in such an arithmetic unit. Subtraction, for example, may require several instructions – a complement followed by a binary addition followed by an addition of 1 (an increment – try the process by subtracting binary 3 from binary 5).

The programmer may also have to provide specific instructions to manage the use of the CARRY bit. The logical instructions which normally can be used to extract part of a word, combine parts, or change single bits or groups of bits must be replaced by complex series of shift, complement, and add instructions. Consider, for example, the sequence of instructions that would be necessary to invert bits 4 and 5 of an 8-bit word so as to flash two warning lights *on* and *off*. Arithmetic units with such limited capabilities are now found only in older computers and in the smallest and cheapest microprocessors.

Modern Arithmetic-Logic Units

Most modern CPUs, including the most widely used microprocessors (Intel 8080, Motorola 6800, Zilog Z-80, MOS Technology 6502, etc.) as well as the control sections of large computers, have arithmetic units capable of performing a wide variety of arithmetic and logical functions (see Fig. 7 for a diagram). These arithmetic units are called *arithmetic-logic units* or *ALUs* and typically have the following characteristics:

1. They can perform many functions in a single cycle. Among the common ones are: a) binary addition, b) binary subtraction, c) logical AND, d) logical OR, e) logical EXCLUSIVE OR (1 if one of the two inputs is 1 but not both), f) ones complement, g) two's complement, h) increment (i.e. an addition of 1), i) decrement (i.e. a subtraction of 1), j) clear (i.e. result is zero), k) comparison (i.e., a subtraction is performed and produces status outputs but the data output is the same as one of the inputs).

2. The function that the ALU performs during a particular cycle is selected by a group of inputs called *function inputs*. For example, in the widely used 74181 TTL device, 1 is logical OR, 6 is subtraction, 9 addition, etc.

3. They have direct provisions for accepting status inputs and producing status outputs. The most common input is CARRY IN which transfers a carry from one word of a multi-word result to the next. Status outputs may include OVERFLOW and PARITY as well as the more common SIGN, CARRY (OUT), and ZERO.

Obviously an ALU is much more powerful than a simple binary adder. The programmer will now have many useful arithmetic and logical functions directly available at high speed. The instruction decoding mechanism will be simpler since it does not have to activate a series of units to perform operations; instead, it selects the sources and destinations of the data as before and chooses the arithmetic or logical function by sending the required code to the ALU function inputs.

The ALU may itself have provisions for shifting and masking data. (*Masking* is the process of removing certain bits or groups of bits from a computer word, such as obtaining bits 3 to 5 of an 8-bit word which may represent the status of a switch or some lights.) There may also be a separate mechanism which performs these functions; the separate mechanism is desirable because then shifting and masking may be carried out together with arithmetic or logical operations in a single cycle.

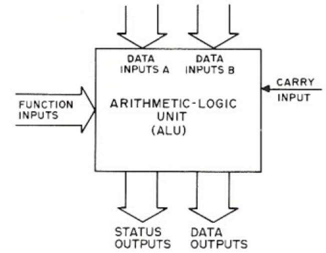


Fig. 7. A modern arithmetic-logic unit showing the various inputs and outputs.

For example, each step in a multiplication or a division requires both an arithmetic operation and a shift. Shifters may allow many complex

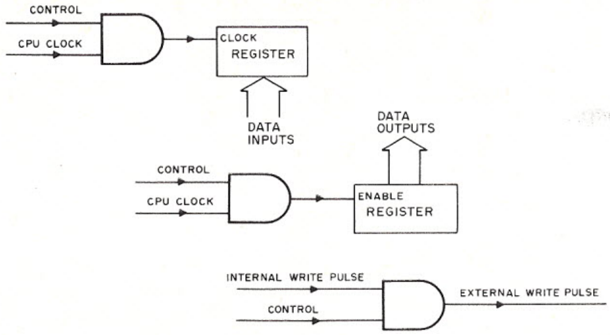


Fig. 8. Using control signals for gating purposes. (a) The control signal determines whether the register will be clocked or not. (b) The control signal determines whether the buffer will be enabled or not. (c) The control signal determines whether the internal write pulse will actually appear externally.

operations including multiple-bit shifts, circular shifts (where the most significant and least signficant ends of the data appear to be connected), arithmetic shifts (where the value of the sign bit remains constant), multiword shifts, and conditional shifts.

Some computers have even more elaborate arithmetic facilities with provisions for direct execution of multiplication, division, square roots, trigonometric functions, and floating point arithmetic. Such arithmetic units have been very expensive and therefore used only in high performance equipment. However, the cost of special-purpose arithmetic circuitry is dropping rapidly and such units should soon find their way into everyday applications. A more complex arithmetic unit results in more powerful instructions and higher execution speeds. The programmer also has less difficulty formulating problems in terms of computer instructions.

Many add-ons are available to increase the arithmetic capability of microcomputers. These include units based on scientific calculator chips, units based on TTL arithmetic-logic units, and units based on high speed multipliers. While microprocessors without hardware multiplication typically require 200 microseconds to multiply binary numbers in software, hardware multiply boards can do the same operation in 2 to 5 microseconds and new LSI chips can do it in 100 nanoseconds. Units based on calculator chips are relatively slow but can replace the software which would otherwise be necessary to perform decimal arithmetic and compute functions such as logarithms, sines and cosines.

Special Arithmetic Facilities

The CPU may use the arithmetic unit to calculate addresses as well as to process data. For example, the CPU must increment the program counter as part of each instruction fetch cycle. The CPU may also have to perform additions or subtractions to calculate the actual values of indexed or relative addresses (the address of the data in these cases is the sum of an index or offset and the contents of the index register or program counter).

An important fact to note is the when the CPU uses the arithmetic area for address calculations, it does not use the status inputs nor does it change the stored status outputs. The instruction decoding mechanism must provide signals that indicate whether or not the status inputs are used and the outputs changed in a particular ALU operation. Obviously we don't want the incrementing of the program counter or the calculation of an indexed address to affect the CARRY, ZERO, SIGN, or other status bits.

Many common microprocessors have a special problem calculating addresses since the addresses are longer than their usual data words. On the Intel 8080, Motorola 6800, MOS Technology 6502, and other widely used processors, the data is 8 bits long while addresses are 16 bits long. Obviously these processors would operate very slowly if they had to use an 8-bit ALU during each instruction cycle to increment the 16-bit program counter or to perform other address manipulations.

Therefore these microprocessors commonly have two arithmetic units – an 8-bit ALU which performs a variety of functions and a 16-bit arithmetic unit which performs a few simple functions. The 16-bit unit may only provide a 16-bit increment or it may provide a few other functions as well. Usually the programmer can also use the 16-bit arithmetic unit to quickly perform a few 16-bit calculations with addresses or 16-bit data.

Computers do virtually all their arithmetic in binary since this is the simplest and most efficient method. However, some arithmetic units either perform decimal arithmetic directly or produce the corrections needed to change the results of binary arithmetic to decimal. Either the programmer or the CPU (through instruction decoding) may be able to select a decimal mode of operation. Decimal addition is the easiest operation to implement; decimal subtraction is somewhat more difficult.

The complexity of the arithmetic unit has a large effect on how easy a computer is to program. A simple arithmetic unit requires the programmer to divide most operations into several instructions and greatly reduces the throughput of the computer. A complex arithmetic unit allows the CPU to have powerful instructions which directly perform required operations. LSI developments are now resulting in even more elaborate arithmetic units which are more flexible and can provide a wider variety of functions. We should expect to see continuing improvements in the arithmetic capabilities of large and small computers.

Instruction Handling Area

The key to any central processing unit is the area that handles the instructions. The fetching, decoding, and execution of instructions is the basic activity of the CPU and of the entire computer. The instruction decoding process must provide the control signals which direct the other parts of the CPU and the other sections of the computer.

Each instruction cycle starts out in the same way; the CPU places the contents of the program counter on the address bus and then places the data obtained from the memory in the instruction register. This process is an *instruction fetch cycle*. We should note that the fetching of instructions is purely an overhead function; the computer does not perform any useful work until it decodes the instruction. The existence of this overhead function means that, however fast a computer may seem to be, it can always be replaced (for any particular task) by faster special-purpose circuitry which performs a single task and therefore does not have to spend time fetching and decoding instructions. The advantage of the computer is not speed but flexibility.

Decoding Instructions

A single instruction will usually be decoded into many separate actions. Bits or groups of bits in the instruction (called *fields*) have particular functions among which are:

1. Choosing which operation the arithmetic-logic unit will perform. Choices may include addition, subtraction, logical functions, increment, decrement, and shift.

2. Selecting sources and destinations for the arithmetic-logic unit. The sources and destinations may be registers or memory locations.

3. Determining whether or when status flags will be changed or used in the operation. The common flags are CARRY, ZERO, and SIGN.

4. Deciding how data addresses are to be fetched, calculated, and used. The address determination may require indexing, relative addressing, indirect addressing, or other methods. The CPU may have to fetch additional words from memory and update pointers or indexes.

5. Selecting control signals to be generated. These signals may include write pulses, timing signals, strobes, and acknowledgments.

6. Choosing operations for separate functional units such as shifters.

7. Providing control signals for registers, buffers, and flip-flops and initializing and clocking internal counters.

8. Providing select inputs to multiplexers and demultiplexers.

Although the CPU represents only a small part of the size and cost of most computer systems, it provides the brains for the entire computer and its capabilities determine the capabilities of the system.

The CPU directs the arithmetic-logic unit through the function inputs as described previously. Many of the other functions involve logical AND gates. The instruction decoding mechanism often provides gating signals – if the gating signal is zero, the AND gate inhibits activity (since zero ANDed with anything is zero); if the gating signal is one, the AND gate passes the other signal which may be a level, a single pulse, or a series of pulses.

Fig. 8 shows some examples of gating signals which allow or inhibit clocking, enabling, or the production of external control signals. A typical example of a function that might be controlled by a gating structure is a write pulse – the internal circuitry of the CPU may generate a write pulse during every cycle but the gating signal is ANDed with the pulse and normally serves to inhibit it. An instruction that involves writing data memory will make the appropriate gating signal one for a specified period of time, thus allowing the write pulse to pass through the AND gate and appear at an output pin.

The instruction decoding may be performed by ordinary circuitry such as gates, flip-flops, counters, multiplexers, demultiplexers, and decoders. A CPU which decodes instructions in this way is said to be *hard-wired*. Most older computers and simple microprocessors (like the Intel 4040) are hard-wired. ■

References

Blakeslee, T.R., *Digital Design with Standard MSI and LSI*, Wiley, New York, 1975.

Osborne, A., *An Introduction to Microcomputers*, Chapter 4, Adam Osborne and Associates, Berkeley, CA, 1976.

Rosin, R.F., "Contemporary Concepts of Microprogramming and Emulation," *Computing Surveys*, Dec., 1969, pp. 197-212.

Salisbury, A., *Microprogrammable Computer Architectures*, American Elsevier, New York, 1976.

Weitzman, C., *Minicomputer Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

Have you written Software for your Altair[T.M.] Computer?

The Altair 8800 computer was the first micro produced for the general public and remains number one in sales, with more than 8,000 mainframes in the field. The wide acceptance of the Altair computer and its rapid adaptation to many diversified applications has truly turned the dream of the affordable computer into a reality.

Yet the machine itself, remarkable as it is, represents only the beginning. The right Software, tailored to meet a user's specific requirements, is a vital part of any computer system. MITS wants to insure that Altair users everywhere have the best applications software available today and in the future. For this reason, a new MITS subsidiary, the ALTAIR SOFTWARE DISTRIBUTION COMPANY, has been formed. Its purpose: to acquire the highest quality software possible and distribute it nationally through Altair Computer Centers.

That's where you come in. The ASDC will pay substantial royalties to the originators of all software accepted into the ASDC library. If you have written business, industrial or commercial use software for the Altair 8800, ASDC wants to hear from you. It is the aim of the ASDC to stimulate and reward creativity in producing useful software that makes those dreams of "computers for everyone" come true. The ASDC will select only software that measures up to its high standards for system design, coding and documentation. The software will then be further documented and distributed through Altair Computer Centers around the country.

For more information on how to submit software to the ASDC, ask your Local Altair Computer Center for an ASDC Software Submittal Packet or contact the ALTAIR SOFTWARE DISTRIBUTION COMPANY.

A subsidiary of MITS
ASDC
ALTAIR SOFTWARE DISTRIBUTION COMPANY
3330 Peachtree Road, Suite 343 Atlanta, Georgia 30326 404-231-2308

see next page for a listing of Altair Computer Centers

ALTAIR COMPUTER CENTERS

BEAVERTON, OR 97005
8105 SW Nimbus Ave.
(503)-644-2314

BERKELEY, CA 94710
1044 University Ave.
(415)-845-5300

SANTA MONICA, CA 90401
820 Broadway
(213)-451-0713

DENVER, CO 80211
2839 W. 44th Ave.
(303)-458-5444

ALBUQUERQUE, NM 87110
3120 San Mateo N.E.
(505)-883-8282;883-8283

TUCSON, AZ 85711
4941 East 29th St.
(602)-748-7363

LINCOLN, NB 68503
611 N. 27th St.
Suite 9
(402)-747-2800

LITTLE ROCK, AR 72206
2412 Broadway
(501)-371-0449

TULSA, OK 74135
5345 East Forty First St.
110 The Annex
(918)-664-4564

HOUSTON, TX 77036
5750 Bintliff Drive
(713)-780-8981

RICHMOND, VA 23230
4503 West Broad St.
(804)-335-5773

SPRINGFIELD, VA 22150
6605A Backlick Rd.
(703)-569-1110

CHARLESTON, W. VA. 25301
Municipal Parking Building
Suite 5
(304)-345-1360

EAGAN, MN 55122
3938 Beau D'Rue Drive
(612)-452-2567

ANN ARBOR, MI 48104
310 East Washington Street
(313)-995-7616

WINDSOR LOCKS, CT 06096
63 South Main Street
(203)-627-0188

PARK RIDGE, IL 60068
517 Talcott Rd.
(312)-823-2388

ST. LOUIS, MO 63130
8123-25 Page Blvd.
(314)-427-6116

NASHVILLE, TN 37203
1600 Hayes St.
Suite 103
(615)-329-1979

BURLINGTON, MA 01803
120 Cambridge St.
(617)-272-8770

ALBANY, NY 12211
269 Osborne Road
(518)-459-6140

NEW YORK, NY 10018
55 West 39th St.
(212)-221-1404

ATLANTA, GA 30305
3330 Piedmont Road
(404)-231-1691

TAMPA, FL 33614
5405 B Southern Comfort Blvd.
(813)-886-9890

ASDC

3330 Peachtree Road, Suite 343
Atlanta, Georgia 30326

COMPUTER HOBBYISTS

WANT SATISFACTION?

We give expert advice and instruction so you can choose the right hardware and software and know how to use it.

We Stock • IMSAI • Lear Siegler Tarbell • Seals • Cromemco Processor Technology • TDL Polymorphic • Icom • and Compucolor.

We Supply Parts, Service and Education.

Our Software Includes Apple, Zapple, Disk Basic, Macro-Assembler & Word Processor.

Send for your FREE 8080 Reference Card.

THE COMPUTER MART OF NEW JERSEY
501 Route #27, Iselin, N.J. 08830
(201) 283-0600
Store Hours: Mon. thru Sat. 10 AM to 6 PM
Evenings: Mon. and Thurs. til 9 PM
Master Charge—BankAmericard Honored

RS 232 INTERFACE CONNECTORS

25 CONTACT

	1-9	10-49	50-99
DB25S (socket)	3.90	3.25	3.00
DB25P (pin)	2.65	2.20	2.00
DB51226-1 (clamp)	1.60	1.30	1.00

We stock a full line in this family of connectors.

9, 15, 25, 37 and 50 contacts and coaxial units and a full stock of accessories, coax contacts hoods, shells, screw locks, sliding locks, etc.

—We have "data phone" types—

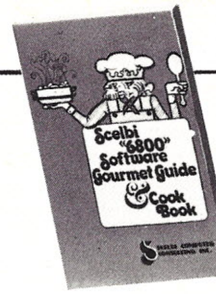
ORDERING INFORMATION:

California residents add 6% sales tax
All orders prepaid
Foreign orders — U.S. funds only

Orders under $15.00 add $1.00 shipping and handling. Prepaid orders over $15.00, we will pay freight.

A-OK ELECTRONICS
5855 W. Centinela Ave.
LOS ANGELES, CA 90045
(213) 670-2266
TELEX 65-3438

only
$9.95 ppd



Now you can cook-up hot programs on your "6800"

A gourmet's delight of practical "how to" facts, including description of "6800" instruction set. How to manipulate "6800" stack. Flow charts. Source listings. Routines for multiple precision operation. Programming time delays for real time applications. Random number generators. Completely assembled floating point math program. Input/output processing for basic I/O programming through interrupt processing. Code, numeric conversion routines. Real time programming. Search/sort routines. Plus many more finger-lickin' goodies.

Order your copy of **Scelbi's "6800" Software Gourmet Guide & Cook Book** today! Only $9.95 ppd. Bon appetite!

SCELBI COMPUTER CONSULTING INC.
1322 Rear Boston Post Road
Milford, CT 06460 • (203) 874-1573

ALDELCO COMPUTER CENTER NOW OPEN

Kits, Books, Boards, Magazines
Special 2102L1 8 for $17.50. We stock OK Battery Operated Wire Wrap tool $34.95, OK Hand Wire Wrap Tool $5.95. 7400 ICs CMOS, Timers PPL's. All kinds of transistors, rectifiers, and diodes.

Plus other electronic parts.

ZENERS
1N746 to 1N759 400 Mw ea .25 1N4728 to 1N4764 1 w 28

C106B SCR	$.65	CA 3028A Dif. Amp.	$1.50
MPSA14	.90	LM301 Op Amp	.55
2N3055	.99	LM309K Volt Reg.	1.10
MPF102 FET	.45	LM380N Audio Amp	1.75
2N3904 or 2N3906	.25	NE540L Power Driver	5.95
2N5496 or 2N6108	.70	NE561B PLL	4.95
MJE340 (2N5655)	1.10	NE562B PLL	4.95
40673 RCA FET	1.55	NE565A PLL	2.50
741 or 709 14 Pin DIP	.25	LM709 Min DIP Op Amp	.45
555 Timer	.75	LM741CE T05 Op Amp	.45
556 Dual 555	1.75	14 or 16 Pin IC Sockets	.30
200 Volt 25 Amp Bridge	1.50	We have Wire Wrap Sockets and	
1N914 - 1N4148	15 for .99	Wire Wrap Wire — 50 feet $1.98.	
1N34 - 1N60 - 1N64	10 for .99		

Send stamp for our catalogue.
Open Mon thru Sat 9 AM—5 PM
Wed till 9PM.

We quote on any device at any quantity. Min. order $6.00. Out of USA send certified check or money order. Add 5% for shipping.

ALDELCO
2281A Babylon Tnpk Merrick, N.Y. 11566
(516) 378-4555

Mark Borgerson
325 NW 9th #3
Corvallis OR 97330

Only Five Senses

... add a few with this converter

I love articles like this. Some of the biggest kicks in this hobby come from watching the computer control things or act upon inputs from the outside world in some way. After reading Mark's article I'd be willing to bet that the Lunar Lander program played at his house is probably the most exciting on the block! — John.

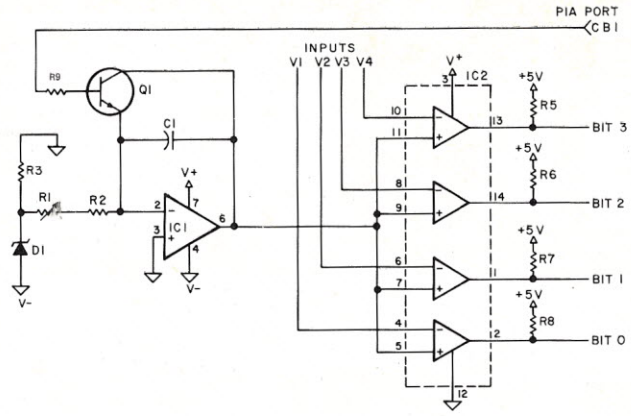
Like many other microcomputer owners, I was bitten by the "analog input bug" almost as soon as my system was up and running. This particular disease had about a 6-month incubation period while I checked on the cost of 8, 10, 12 and even 13-bit analog-to-digital converters (A/Ds). The price tags in the tens of dollars and the need for external precision reference voltage generators, clocks and signal conditioning delayed the cure to the analog input bug for some time. I finally got tired of waiting to see if anyone was going to market a simple A/D board for the SWTPC 6800 and decided it was time to build my own. What I wanted was a simple (meaning inexpensive) A/D converter that would respond to input voltages of from zero to five volts (a digital voltmeter). Several designs using 555 timer circuits have appeared in other magazines, but these converters respond to resistance or potentiometer position — a factor I found too limiting. The other requirements for my design were those normal to the hobbyist who is not an electrical engineer: simple design, minimum cost, and minimum parts count. The simplicity of the resulting circuit surprised me, and I hope it will cure the analog input bug as well for you as it did for me.

Once you finish constructing a project like this, the urge to put it to use is immediate and overwhelming. The first thing I did was put a potentiometer across a 5-volt supply and sit there happily watching the displayed output voltage change as I twisted the knob on the pot. After a few minutes this grew to be somewhat less than exciting. The next thing I did was to modify the input section of a very simple Lunar Lander game so that the amount of fuel to burn was set by the position of the potentiometer. I immediately found that this modification changes the whole character of the game. You can no longer sit quietly and ponder each move for as long as you wish. That input has to be set right when the computer asks for it — it won't wait for wishy-washy pilots! (The game as I had it set up was awfully tricky and not too well written, so I'll rewrite it for a later article.) The Lunar Lander type games are well documented in a number of books (for example, *What To Do After You Hit RETURN* by the Peoples Computer Company ... available from *Kilobaud Magazine*), so get a listing and start on your own modifications. After all, writing your own programs is what personal computers are all about!

The Big Picture

The first step in attacking the analog input problem is to take stock of your resources. I am going to assume your basic weapons are an M-6800 based microcomputer and the input half of a PIA (Peripheral Interface Adapter). In addition to the input port, you will need the CB2 control output from the PIA to control the A/D converter.

The parallel input port is the most obvious resource in this attack on the analog input bug, but we have another strong ally on our side ... *time!* Not time as measured in weeks, days, hours or even seconds; time as measured in computer clock cycles. When it comes to measuring things, computers have a restricted vocabulary. The number of words in their voltage vocabulary is two (the high and low states of digital logic). The basic measure of time in the computer is the clock cycle — or more properly, the number of clock cycles needed to run through a simple counting loop. If we use an 8-bit accumulator to keep track of



IC1	741 Operational Amplifier
IC2	339 Comparator
Q1	2N3904 Transistor
D1	4.7V zener diode
C1	0.1 uF Capacitor (Polystyrene or polypropylene type)
R1	10K-Ohm potentiometer
R2-R3	10K-Ohm ¼W 5% resistor
R4-R9	1K-Ohm ¼W 5% resistor

Fig. 1. Schematic diagram of the A/D converter.

the number of runs through the loop, then we can measure up to 256 increments of time, each one the length of time needed to cycle through the loop.

We now have a computer which can sense only two different voltage levels, but can be programmed to measure up to 256 different time increments. If this seems a little backwards to you, you are absolutely right — we want 256 levels of voltage sensing for the A/D conversion. The problem seems to be to convert from elements of time to elements of voltage (in instrumentation circles this is known as a change in the *Data Domain*). The solution to this problem is a clocked linear ramp generator. This ramp generator has an output, which stays at zero volts until a control signal arrives and then begins to increase its output voltage by a set amount in each succeeding time interval.

Now we are ready for the final step in our analog-to-digital conversion. We have set up a counter which will measure up to 256 increments of time and a ramp generator which will increase a voltage by a fixed amount in each of these time increments. The final element is an analog comparator. This integrated circuit accepts two input voltages, one from the ramp generator and one from the voltage we want to measure, and produces an output that indicates which of the inputs is the higher voltage.

The overall plan of attack goes like this: Use the control output of the PIA to set the ramp generator to zero volts, then let it begin ramping upward; enter a counting loop and in each cycle of the loop check to see if the ramp voltage is greater than the input voltage (indicated by the bit at the input port changing to a 1). If the ramp voltage is less than the analog input, increment the B accumulator and start the loop over. If the ramp is greater than the analog input, save the value in the B accumulator in a special memory location. The value in the B accumulator is directly proportional to the ramp voltage — and thus to the input voltage.

Construction

The instructions so far have been somewhat hypothetical. It's now time to present the real-world hardware and software to implement a simple 8-bit A/D converter. A schematic diagram of the hardware is shown in Fig. 1. The linear ramp generator consists of IC1, transistor Q1 and capacitor C1. The ramp generator is reset to zero volts by applying a logic 1 level to the base of Q1. The ramp voltage begins increasing when the base of Q1 goes to zero volts. The rate of increase is set by the flow of current through resistors R1 and R2. R3 and D1 set a stable -4.7 V reference for the ramp generator. For maximum stability and linearity capacitor C1 should be a polyethylene or polypropylene type. The V+ and V- connections for the circuit may range from ±9 to ±15 volts. The output of the ramp generator is connected to the four noninverting inputs of a 339 quad comparator. This comparator is designed to work from a single power supply and will accurately sense voltages near ground. Each of the four inverting inputs can be connected to a different analog input voltage. The four comparator outputs are connected directly to the four lower-order bits of the parallel input port. The comparator outputs are similar to TTL open-collector outputs and are TTL compatible. The pull-up resistors, R5-R8, are not necessary in all cases, but will add some noise immunity at the input port. If more than four analog inputs are needed, another 339 comparator may be added to produce a total of 8 analog inputs. The total cost for the hardware should not be more than four dollars if you already have the PIA port available.

The Software

The machine-language subroutine to start the ramp generator and count the cycles through the loop is listed in Program A. The format of the listing is similar to the output format of the SWTPC or Motorola assemblers, except that line numbers and assembler directives have not been shown. This is simply because I wrote the routine in machine language, then added the mnemonics and comments later. You don't necessarily have to be able to write routines like this to enjoy your computer, but it certainly does help when you are debugging home brew interfaces.

To make this routine operate properly, you must tell the counting routine

```
1FC0                 COUNT    RMB 1          Save location for count
1FC1    5F           START    CLRB           Clear counter
1FC2    86 34                 LDAA #$34      Load $34 for control low
1FC4    B7 8017               STAA $8017     Store at PIA control
1FC7    B6 8016      LOAD     LDAA $8016     Load from PIA input
1FCA    5C                    INCB           Increment counter
1FCB    85 01                 BITA #$01      Test: bit X high?
1FCD    27 F8                 BEQ LOAD       Bit not set, loop back
1FCF    F7 1FC0               STAB COUNT     Save count
1FD2    86 3C                 LDAA #$3C      Load $3C for control high
1FD4    B7 8017               STAA $8017     Store at PIA control
1FD7    39                    RTS            End of Subroutine
```

Program A. This is the machine-language subroutine which starts the ramp generator and increments the counter until the appropriate bit at the input changes state (0 to 1). The bit mask for the appropriate input must be stored at location 1FCC before entering the subroutine.

```
1   REM DIGITAL VOLTMETER PROGRAM FOR SWTPC 8K BASIC.
2   REM MARK BORGERSON DEC. 14, 1976
3   REM MEMORY LIMIT (LOCATIONS 0044 AND 0045) MUST
4   REM BE SET TO 1FBF BEFORE ENTERING PROGRAM.
5   DIM M(4)
6   DATA 1,2,4,8
7   READ M(1), M(2), M(3), M(4)
15  REM SET ADDRESS OF (USER) TO 1FC1.
16  POKE (103, 31)
17  POKE (104, 193)
20  REM VOLTAGE MEASURING LOOP.
25  FOR I=1 TO 4
30  POKE (8140, M(I))
35  A=USER(0) + PEEK(8128)/255*5
40  PRINT TAB(I*10-9); INT(100*A+.5)/100;
45  NEXT I
50  PRINT CHR$(16)
55  GOTO 25
```

Program B. The PRINT statement in line 50 will cause a "home-up" on TV-typewriter terminals equipped with computer cursor control. On a TTY, the data will be printed four readings per line and will eventually use up a lot of paper.

which input channel you want to measure. This is done by storing the appropriate bit mask at location 1FCC. The bit masks for channels one to four are 01, 02, 04, and 08 respectively. After returning from the machine language routine, the loop count, which is proportional to the input voltage, is stored in the COUNT location and is present in the B accumulator. The count is stored in a specific location to make it accessible using the PEEK command of SWTPC 8K BASIC. In order to relate the count output directly to the input voltage it is necessary to calibrate the ramp generator so that it increases by a known voltage increment on each cycle through the counting loop.

Program B is written in BASIC to make your computer act like a four-channel digital voltmeter. To do this, it samples each of the input channels (using the appropriate bit mask) and scales the resulting number to produce a reading between 0 and 5.00 volts.

Calibration

By applying a known input voltage to all four channels and adjusting resistor R1 until the outputs at the terminal agree with the known input, the A/D converter can be calibrated.

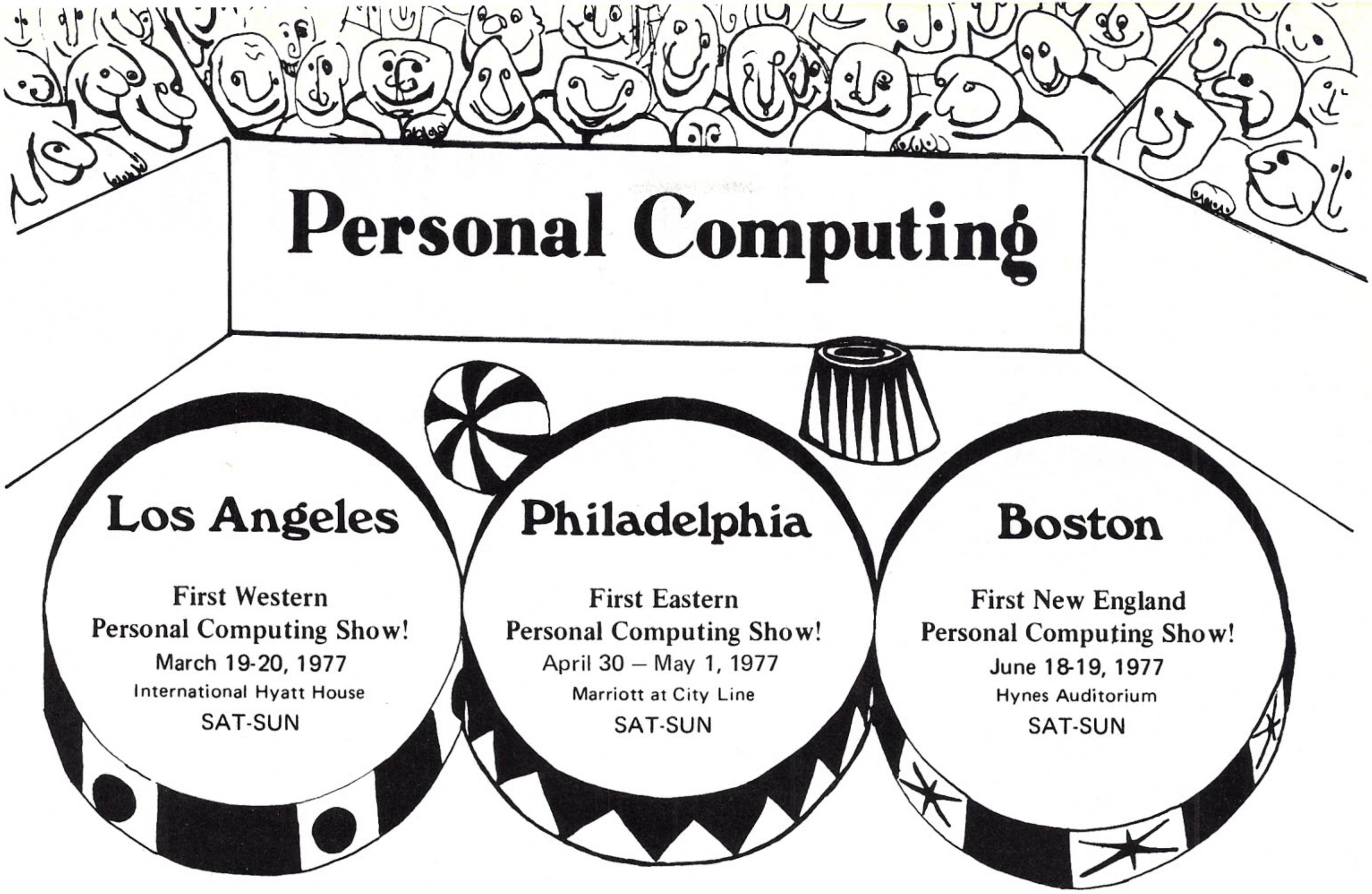
The absolute accuracy of the calibration is dependent on the accuracy with which you know the input voltage, so use the best source you can get your hands on. If you can get a good digital voltmeter, you can use a potentiometer across a 5-volt supply as the voltage source. Simply connect the wiper of the pot to both the A/D input and the DVM input. You can then adjust R1 until the meter and the computer readout agree. In a pinch, you could use a good analog multimeter, but when the circuit is set up for a 5-volt full scale input and is reading about 4.96 volts, the resolution is about 0.02 volts — much better than the resolution of most analog multimeters. Another possible source of a calibration voltage is a common mercury battery. A mercury cell in good condition has an output voltage between 1.35 and 1.37 volts.

In most home computer applications, absolute voltage accuracy is not needed. Most of the time you will need only the ability to compare one voltage with another or to monitor the change in a voltage with time. In both these applications the four-channel circuit described here works very well. I have found that the four inputs match each other to within the 0.4% tolerance inherent in the hardware and software. In other words, when the inputs are all connected to the same voltage and that voltage is varied, the outputs of the four channels always agree to within 0.02 volts. In addition, when the input is set to 4.00 volts, the outputs continued to indicate 4.00 volts for more than an hour without varying by more than 0.02 volts. The long-term stability of the circuit is probably affected most by changes in room temperature which will change the value of R1. In a normal household environment, this should not be a very great problem.

Conclusion

There are many other applications for A/D converters. A short list might include: light sensing with a photocell for control of home lighting, monitoring the output of a motion sensor in a burglar alarm system, measuring room temperatures with thermistors to regulate home heating, and inputs to a simple video graphics system with an inexpensive joystick module.

When you come up with a really good application for A/D converters in the home computer environment be sure to send it in and get it published. Not only will it help pay for that next piece of new hardware, but it will help the rest of us avoid duplicating your hours of frustration and cursing in the debugging stage. ■



Greatest Computer Shows Ever!

Personal Computing magazine is proud to announce that it is sponsoring the first series of regional Personal Computing Shows.

Beginning with the *Western Personal Computing Show* in Los Angeles, and followed by the *Eastern Personal Computing Show* in Philadelphia and the *New England Personal Computing Show* in Boston, **Personal Computing** magazine intends to make everyone aware of low-cost computing.

Other shows are now being planned for the South, Southwest, Canada, and Europe!

Already, invitations have been sent to all the manufacturers in the personal computing field, computer stores, computer clubs and well-known computer experts.

Special areas of the exhibition halls will be set aside for Personal Computing in Education, in the Home, in HAM Radio, and in Small Businesses. These are all first for a computer show.

Seminars and special presentations include: Computer Synthesized Music, HAM Applications, Trends in Microcomputers, Mass Storage Systems, Lemonade Computer Service Companies, The Kitchen Computer, Computers on the Farm, The Small Business System, Software for Fun and Practical Applications, Computer Club Organization, Standards for the Hobbyists, Computer Art, The House Robot, Computer Crime, Software Protection and Future Computing.

In addition, *special tutorial workshops* will cover all aspects of computer hardware, programming in both machine language and higher-level language and applications. Workshops are designed for both beginners and advanced students in the art of personal computing.

We anticipate 150 different exhibits and crowds of up to 10,000 people at each of these shows. Arrangements for the shows are being handled by a professional management company to ensure that everything runs smoothly.

Cost of Registration:

At the door:
- $10 per show (two days)
- $ 6 per One Day Pass

Special Pre-Registration Rates:
- $ 7.50 per show (two days)
- $ 4.00 per One Day Pass

Note: Show tickets and one day passes entitle you to attend all seminars, workshops, exhibits and other events.

Register Now and Save!

Yes, I would like to take advantage of your special, pre-registration rates. I plan to attend the following regional Personal Computing Show(s):

☐ Los Angeles	☐ Philadelphia	☐ Boston
☐ Show (two days)	☐ Show (two days)	☐ Show (two days)
☐ One Day Pass Only	☐ One day pass	☐ One day pass

Enclosed is a check for ____________________

Name ____________________

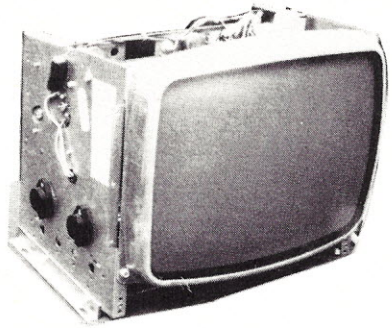
Address ____________________

City ____________________ State & Zip ____________________

Send to: **Personal Computing**, Conference & Exposition Management Co., Box 844, Greenwich, CT 06830.

DELTA ELECTRONICS CO

POST OFFICE BOX 2, AMESBURY, MASS. 01913 Phone (617) 388-4705

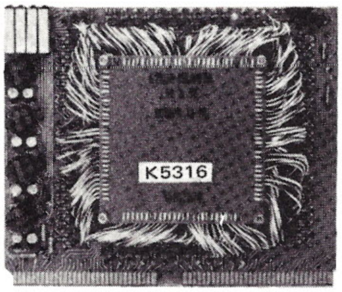


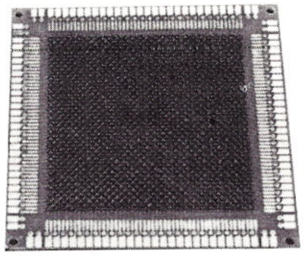
12" MOTOROLA VIDEO TERMINAL

We have a limited number of MOTOROLA VIDEO TERMINALS, which were taken out of service when W.T.Grant went out of business. We have checked them on our own computer. Several work perfectly, and the rest had some vertical roll, possibly our fault. We are selling them AS IS, on a first come first serve basis. Motorola tells us that data is available. Your chance to upgrade your computer at a minimal price.

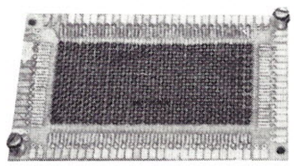
K5484 VIDEO TERMINAL As is $69.00

CORE MEMORY BOARDS





K5486

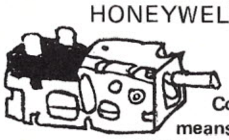


K5446

We have 3 different core memory boards. STOCK NO.K5316 has a total of 10,816 bits, with core drivers and sensors, made by BURROUGHS; comes with data. STOCK NO.K5446 2704 bits, is easily stacked to raise capacity. Core memory only (no driver or sensor circuitry). STOCK NO.K5486 is identical to K5446 except bit capcity is double, 5408 bits. All boards are in limited quantity.

STOCK NO.K5316	BURROUGHS MEMORY BOARD 10,816 bit .	7½"x9"	12.95	2/24.00
STOCK NO.K5446	2704 bit memory board	2¾"x4¼"	4.95	2/8.00
STOCK NO.K5486	5408 bit memory board	4¼"x4¼"	6.95	2/12.00

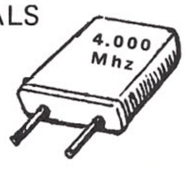
HONEYWELL HUMIDITY CONTROLLER



Made by HONEYWELL for automatic control of humidifiers or de-humidifiers. Control range 10% RH to 60% RH, sensing by means of a nylon ribbon. Snap action switch with 240V AC contacts. Ideal for home lab or greenhouse.

3½ x 2¼ x 1¼. STOCK NO. B6263 1 lb. $4.95 each, 3/14.00

CLOCK and TIME BASE CRYSTALS

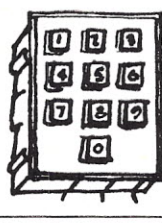


STOCK NO.K1000 4.000 Mhz Crystal HC/18 holder, pin leads

STOCK NO.K1008 5.000 Mhz Crystal HC/18 holder, wire leads.

Either crystal $3.95 2/7.00

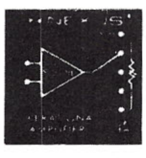
DATA INPUT / TOUCH TONE ENCODER BOARD



The K5151 DATA INPUT/TOUCH TONE ENCODER BOARD is a heavy duty, 10 digit board, made by AUTOMATIC ELECTRIC CO., manufacturers of telephone equipment. Each button operates 2 individual SPST switches, for maximum flexibility in encoding. Each board indiv-idualy packec in foam. Usefull for data input, repeaters, touch tone devices, calculators etc.

STOCK NO.K5151 10 Digit keyboard $3.50 ea. 2/6.00

Philbrick/Nexus OPERATIONAL AMPLIFIER



Philbrick/Nexus Model SQ-10a is a high gain, high stability general purpose op amp. It operates over the temp. range of – 25°C to +85°C. Both inputs & output are fully protected against shorts to ground or the power supply. It may be used open loop as a voltage crossing detector. 1 MHz unity gain bandwidth. Open loop gain 60,000 min. Slew rate 1.1 V/ sec. min. 100 nA. max input bias current. Nominal power supply ±15V 1 1/8" x 1 1/8" x ½"

Current factory price is $12 STOCK NO. B4539 With data $1.75 each, 4/6.00

100 WATT ISOLATION TRANSFORMER

100 watts, continuous operation. 115 volts to 115 volts.
2 7/8"x3 3/8"x3½" 7 Lbs.
STOCK No.K9971 $5.95 ea. 2/11.00

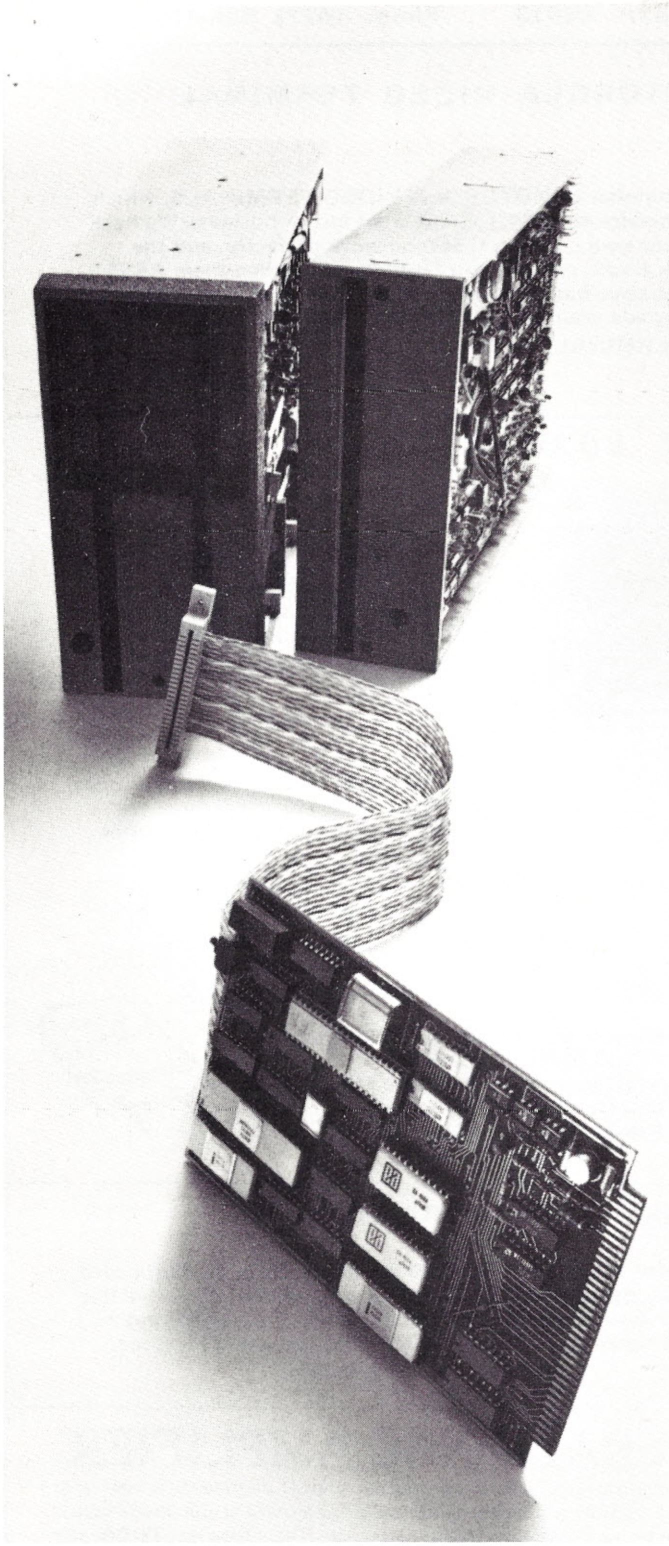
HIGH CURRENT TRANSFORMER

This transformer will deliver 1 of the following : 15 V.@ 15 A., 12 V. @ 15 A., 9 V. @ 10 A, 6 V. @ 8 A.or 3 V @ 6 A. 7 lbs.
STOCK NO.K6481 $8.95 ea. 2/16.00

Send for our latest free catalog. We welcome Master Charge & BankAmericard orders. Minimum credit card order $15.00. Please include ALL numbers for proper processing. Include sufficient postage. Excess is refunded. Minimum order $5.00

The PerSci 1070 controller shown with single and dual disk drive units. This controller is based on an 8080 and can call files by name.

Douglas Hogg
36 Calle Capistrano
Santa Barbara CA 93105



Floppy Disks

... What's the real story?

Doug has put together one of the most complete buyers guides for floppy systems we've seen in quite some time. He's broken it up into three sections: the available controllers, the drives, and the software. With the new minifloppies coming onto the market, the prospect of more people finally being able to afford disks for their systems is looking better and better. — John.

One of the most pressing needs for the computer hobbyist is for mass storage devices. This is not just for supplementing existing memory capacity but also to provide a nonvolatile means of storing programs. The simplest method of bulk storage is the audio cassette recorder. It is easy to interface, cheap, and reasonably reliable, but it has several shortcomings. The first is speed — most cassette units operate at a maximum of about 30 characters per second and thus require about five minutes to load 10K of memory. The one exception to this is the Tarbell interface which can run up to several hundred characters per second. Even this unit suffers from the disadvantage of not being computer controlled. For instance, the computer cannot rewind the unit or do a high speed search for a particular program. Paper tape is generally faster and

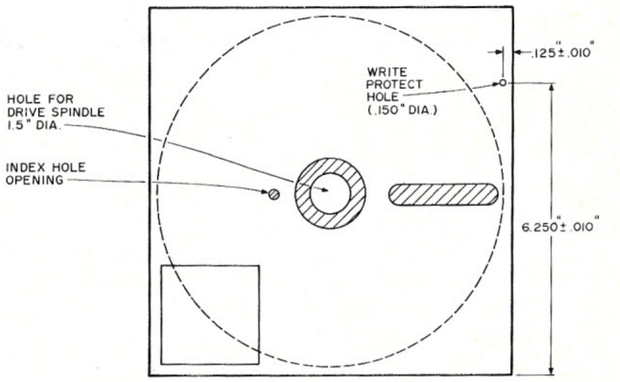
more convenient than cassettes but it lacks the flexibility of magnetic recording. Digital cassettes, such as the ones from the Digital Group, are a partial solution to the problem. They are computer controlled, will do a high speed search, have a reasonable data transfer rate (about 800 characters per second) and are reasonably priced, about $375 for two drives and a controller. Their one disadvantage is the access time, which may be as much as 40 seconds depending on the position of the tape. There is a better solution, however. How about a device which can store 256K bytes, access any part of this storage in less than a second, and transfer information into the computer at a rate of 30,000 bytes per second? The device which can do all of this is the floppy disk. The fast access time and high data transfer make it practical to do a number of tasks, such as overlays and storing of intermediate results generated during the execution of a program, which take too long, even with digital cassettes, to be reasonable. There are six component pieces to a floppy disk system: the disk, the disk drive, the drive electronics, the controller, the computer interface, and the system software. These parts are discussed individually below. After this discussion, specific commercially available components are reviewed.

The Floppy Disk (also referred to as "the diskette")

Basically the floppy disk is a large round piece of recording tape enclosed in a protective envelope. The standard disk is shown in Fig. 1. The protective envelope is eight inches square with cutouts for the drive spindle, recording head and index position hole. The disk itself is made from 0.003" mylar covered with a thin layer of magnetic oxides.

In nearly all of the drives, the data is recorded in IBM standard format. In this format, the disk is divided into 77 tracks. The tracks are in concentric circles starting near the edges of the disk. That is, one revolution of the disk, with the head fixed, covers one track, as shown in Fig. 2. Each track is further divided into sectors, the IBM format containing 26 sectors per track with the count starting at the index hole as shown in Fig. 3. The sectors may be marked either with information recorded on the disk (Fig. 3) or with actual sector holes punched in the disk (Fig. 2). These two methods are called soft sectoring and hard sectoring respectively. (An index hole is used as a starting reference in both cases.) Each IBM format sector is further divided, as shown in the lower part of Fig. 3, into four sections – two address markers, an ID field, and the data field. The first address marker identifies bytes between this mark and the next address marker as being the address portion of the sector. The ID field contains the sector and track number which are written during the initialization of the disk. The second address marker may be one of two types, the first type identifies a good record while the second type identifies either a bad sector or a sector containing a deleted record. The last section contains 128 bytes of data. A more detailed division of the sectors is shown in Fig. 4. The ID section contains seven bytes, 1) ID, 2) track address, 3) zero, 4) sector address, 5) zero, 6 and 7) two CRC (cyclic redundancy check) characters used for error detection. Thus even if the drive stops at the wrong track and wrong sector there is enough information for the controller to correct the positioning error while the CRC characters allow the controller to see if the error was in reading or positioning. The data section consists of an ID byte, 128 data bytes, and two CRC check characters for a total of 131 characters. There are 33 bytes in the first address marker and seventeen in the second address marker. So to record 128 bytes of data we use a total of 128+3+7+33+17 = 188 bytes. Since an unformatted disk holds about 400K bytes, a formatted disk can hold (128/188)*400K = 250K bytes. The IBM format is not particularly efficient but it allows for very reliable data reading and writing.

Fig. 1. A floppy disk in its protective envelope.



A particularly attractive feature of the floppy disks is the low cost of the storage medium – about $6 each in quantities of ten.

The Floppy Disk Drive

The floppy disk drive unit consists of a positioner for the disk envelope, a drive motor to turn the disk, a stepping motor or similar positioner for the read-write head, and a light and detector for the index hole in the disk. These features are shown in the block diagram in Fig. 5, which is taken from the Shugart SA800 manual. The disk sits on a drive cone (not shown in Fig. 5) which fits into the drive hole in the center of the disk. The cone may be either driven directly by the motor or, more commonly, by a connecting belt. The head is positioned over the proper track by one of two methods – either a stepping motor driving a positioning screw or a voice coil type positioner. The voice coil units have access times as much as five to ten faster than the stepping motor but they require more electronics to accurately position the head. The read-write head and pressure pad (positioned on the opposite side of the disk) generally do not touch the disk until information is to be read or

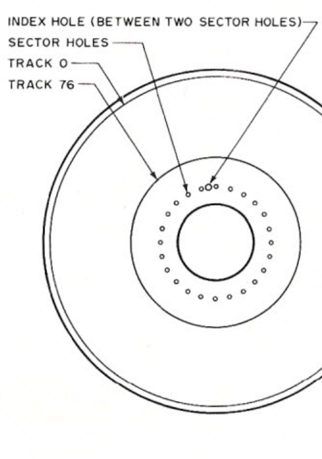


Fig. 2. A hard-sectored floppy disk showing the locations of the tracks.

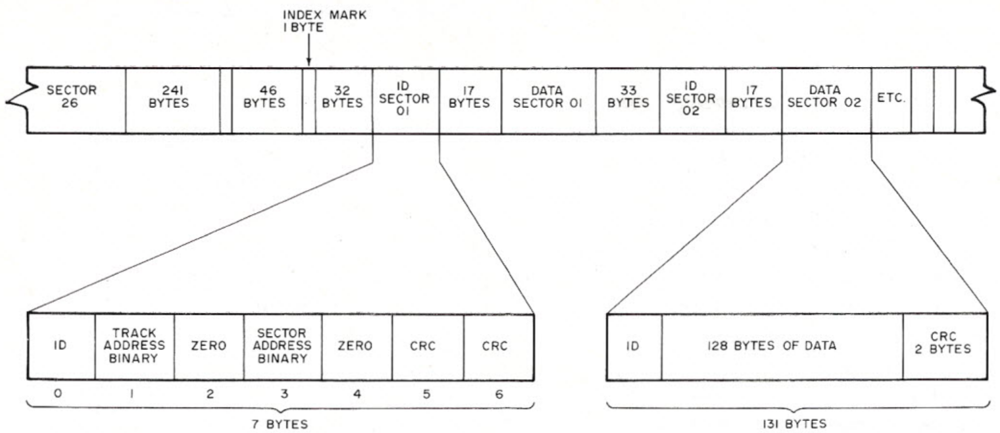


Fig. 4. The IBM data format.

written. This head-loading time varies in different drives from 30 to 75 msec. The index light and detector are positioned over the index hole cutout and tell the controller when sector 1 is in line. Some units have a write protect option which consists of an LED and detector positioned over a small hole near the edge of the floppy disk envelope. This feature functions like the write protect tab on an audio cassette. That is, if there is a write protect hole, the disk cannot be written on.

The Floppy Disk Drive Electronics

The drive electronics consist of the read circuits, write circuits, the motor control circuit and the head position and loading circuits. The drive electronics are typically interfaced to the controller which generates the signals (drive select, write data, read data, clock step, motor on, etc.) to operate the disk. While the drive electronics and the controller could in principle be on one board, they generally are not. Since the physical construction of the drive units may require them to have very different control signals (a good example is the stepper head positioner versus the voice coil positioner), a controller which interfaced directly with a drive would only be useable with that one drive. By having a drive electronics board between the drive and controller it is possible to produce a general purpose controller board.

The Controller

The controller is a processor which translates higher level commands from the host computer into instructions appropriate to the disk drive electronics. Typical instructions to the controller, given in the form of two-byte commands, are: Reset (resets the controller and all floppy disks in the system), Seek (steps the head to a specified track and veries the correct position), Read (reads a sector of data (128 bytes) from a specified sector and verified it with the CRC data), READID (reads the next sector ID information), Write (writes a sector of data (128 bytes) with the normal address mark in the specified sector), WRDEL (writes a sector of data with a deleted address marker in the specified sector), Format (writes address markers, gaps, and data on an entire track according to the IBM format), and Status (reads the drive status of the active drive). Many of the controllers are now based on microprocessors. Shugart, SMS, and GSI use bipolar microprocessors while PerSci uses an 8080. The controllers are discussed individually below.

The Computer Interface

The output of the controllers is generally in the form of bidirectional data lines and several status lines which then need to be interfaced to the host computer. In some cases this may be as simple as plugging the proper card (supplied by the controller manufacturer) into the computer backplane. Systems which have the controller and the interface on one card (such as North Star Computers products) are the most convenient to use but limit themselves to only one type of machine — which is fine if you have that type of machine. Generally the controller and interface are kept separate for the same reason, adaptability, as are controllers and drive electronics.

The interface may be done in two ways. The first, and simplest, is to input the data and output data and commands through standard parallel interfaces. This generally requires two output ports (data and commands) and two or three input ports (data and one or two for status). The second way to interface the controller is to use direct memory access (DMA) in which the controller takes control of the bus and either reads from or writes into the computer memory directly. This approach, which cuts down on the software overhead considerably, is used by Digital Systems and the software from Digital Research.

The Software

This is the most crucial part of the system as the disk and associated parts are not useful without the proper driving software. Actually any computer system is useless without proper software it is just that with a disk based system you have a much more powerful useless system. The purpose of the software is to keep track of what is stored on the disk and where. In addition, the software raises the level of involvement of the operator from calling for information by track and sector to calling programs or data by name and having the computer find which files are being asked for and then inputting (or outputting) them and checking for reading (or writing) errors. A good disk operating system will also contain routines for moving programs from device to device, an assembler, and perhaps a disk accessing BASIC. This last point is very important as having the disk and a file oriented operating system is of no help if your assembler and/or BASIC cannot com-

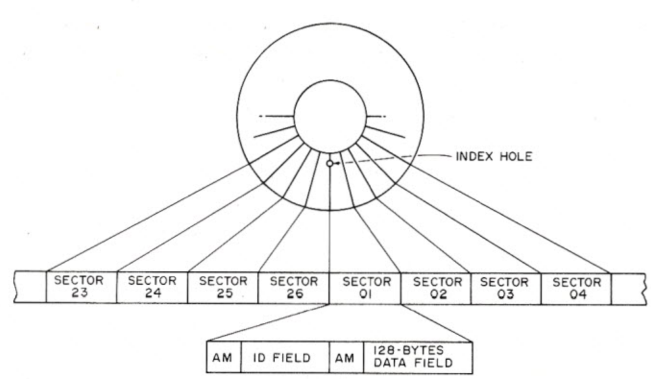


Fig. 3. The IBM format for a soft-sectored disk.

municate with the disk. As an example, standard Altair BASIC has no provisions for supporting high speed peripherals and thus has no way, without modifications, of using the power of the disk to store programs. The disk versions of BASIC have provision for calling in programs from the disk by file name. The same method is used for dumping existing programs on the disc. Patching an existing operating system to access a disk is not a particularly pleasant task and fortunately there appear to be some good disk operating systems available for the 8080 and 6800 even now.

This is a good time to bring up the matter of speed of the disks. The literature all says prominently that the data transfer rate is 250K bits per second (about 30K bytes per second), a rate which could fill a 10K memory in about one third second (or about one half second including the average access time). However, the ICOM advertisements say that you can load an 8K program in less than seven seconds, a data rate of about 1100 bytes per second. Why the difference? There are several reasons, latency, head loading time, settling time, and track-to-track access time. Using a controller which requires extensive software support will be much slower since if you do not get ready for a sector when it comes by you must wait one turn of the disk (167 msec) for it to come around again. Thus if your system is not fast enough to prepare for sequential sectors, the data rate can be improved by having consecutive data stored in spaced out sectors. The speed of the TCH controller can be increased from an average of about 750 bytes per second to as much as 3000 bytes per second by putting the consecutive data in sectors spaced eight apart. Even if the software is fast enough to transfer consecutive sectors (the DMA systems may be this fast), some care is required to optimize data rate. The rate can be as much as doubled for large programs, ones which take more than one track to store, by offsetting the start of the data on each track. For instance, if a large program starts at sector 1 on track 10, fills the rest of track 10 and continues at sector 1 on track 11, the disk will have to wait one revolution for the sector to come by again since the track-to-track move time and settling time may be several sector times. If the program had continued at sector 8 or so, one less disk revolution would be required, increasing the average data transfer rate considerably. So depending on the quality of the software and the quantity of hardware, the average data transfer rate may vary from 750 bytes per second to perhaps as much as 10K bytes per second. The average rate will never reach the 30K bytes per second instantaneous rate because of the times required by the mechanical parts of the system.

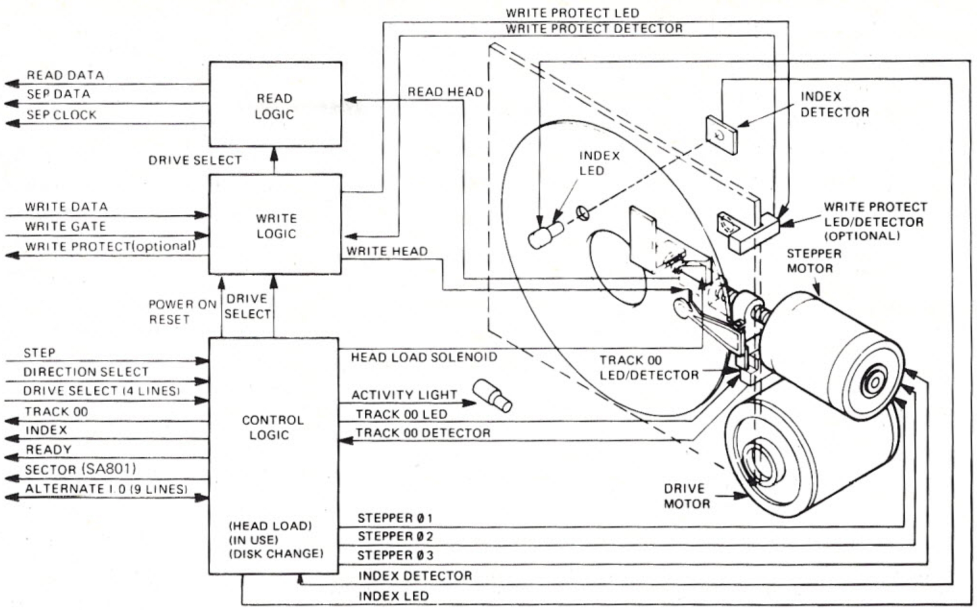


Fig. 5. The construction of the floppy disk drive unit.

What is Available?

This section consists of a discussion of the specific equipment and software available from different manufacturers. The floppy disks (diskettes) themselves are available from all drive manufacturers and many supply places. Since they are all very similar and of high quality, we will not discuss particular brands except to say that the IBM disks appear to be the most popular. The floppy disk drives and the drive electronics come as one package so they are discussed together. Controllers and interfaces are discussed together since the specific interfaces offered are offered with particular controllers. The two available software packages are discussed in the separate software section. In considering what equipment is available, it is good to keep in mind that unless you are the intrepid type who enjoys challenges, and there are many of us left, to ensure compatibility of the various portions of the system it is perhaps wisest to purchase an entire system including the software from one source, or sources which claim to be compatible with each other.

Floppy Disk Drives

There are several standard specifications common to most of the disk drives: Number of disks — 1; Capacity per disk — 3.2 million bits unformatted; Head — Read-write with erase on either side to ensure a separation between the tracks; Rotational speed — 360 rpm; Data transfer rate — 250K bits per second; The description of each unit will include only the unique features or the variation from these standard specifications.

GSI-105, 110 (General Systems International, Inc., 1440 Allec Street, Anaheim CA 92805). Price: about $550 (105). Head engage time: 35 msec. Average access time: 156 msec. Power: 120 V, 24 V @ 1.8 Amps, 5 V @ 1.3 Amps, -5 V @ 0.1 Amps. Special features: fast access time — GSI-110 will record at double density which allows over 500K bytes per disk and 500K bits per second transfer rate.

MDD 50, 51, 52 (General Systems International) Price not available. These newly introduced units use the mini-floppy as do the Shugart SA400. However, these units offer a capacity of 65K (MDD 50), 131K (MDD 51), and 232K (MDD 52) bytes formatted with data transfer rates of 125K, 250K, and 500K bits per second respectively. The models 51 and 52 use double density recording and the model 52 turns at 600 rpm, twice the 300 rpm of the models 50, 51, and

SA400. No other information is available at this time.

Orbis 77 (Orbis Systems Inc., 14251 Franklin Ave., Tustin CA 92680) $704. Head load time: 30 msec. Track to track access time: 6 msec. Power: 120 V, 5 V @ 1 Amp, 24 V @ 1.2 Amps.

PerSci 70, 270 (PerSci Inc., 4087 Glencoe Ave., Marina Del Rey CA 90291) $625 (70), $1000 (270). Head engage time: 40 msec. Track-to-track access time: 10 msec. 76 track seek: 100 msec. Power: 5 V @ 4 Amps, -5 V @ 0.14 Amps, 24 V @ 0.46 Amps. Special features: Linear voice coil motor for positioning the head. The model 270 is one drive unit that handles 2 disks simultaneously.

Pertec FD400, FD500 (Pertec, 9600 Irondale Ave., Chatsworth CA 91311). Price not stated. Head loading time: 40 msec, track-to-track access time: 10 msec. Average access time: not stated. Power: FD400 24 V @ 2 Amps, 5 V @ 1 Amp, -5 V @ 0.3 Amp. FD500 120 V.

Shugart SA800/801 (Shugart Associates, 435 Indio Way, Sunnyvale CA 94086) Price for basic model is $580 in quantity of one. Head load time: 35 msec. Track-to-track access time: 10 msec. Average access time: 260 msec. Power: 120 V, 24 V @ 1.3 Amps, 5 V @ 0.8 Amp, -5 V @ 0.05 Amp.

Shugart SA400 (Shugart Associates) Price $390. This unit differs considerably from the preceding units in many respects besides price. First, it uses a small disk called a minidiskette™ which is 5.25" square compared to the 8" square standard disk. The drive itself is very small (3.25" high, 5.75" wide, and 8" deep) and uses very little power (15 Watts typical). It is, however, slower and has much less capacity than the standard disk. This drive and the companion controller, the SA4400, are intended for cost sensitive applications, a description which fits most of the hobby market. Capacity per disk in a modified IBM format: 80.6K bytes. Format: 18 sectors of 128 bytes per track, 35 tracks per disk. Transfer rate: 125 kbits per second (125 kilobaud). Rotation speed: 300 rpm. Head load time: 75 msec. Track-to-track access time: 40 msec. Average access time: 463 msec. Most of the specifications here are slower than the standard floppies, a feature which contributes to the low cost and probable reliability.

Other common units about which we have received no information are: Innovex 220, Remex RFD7400, CDC 9400, and CALCOMP 140.

Controllers

While the various drives differ only slightly in features and price, the same is not true for the controllers. The basic difference is how much is expected from the host computer by the controller, with the super simple interface designed by TCH requiring total support from the host to the PerSci 1070 which contains its own 8080 and can access files directly by name. The following discussion lists the various controllers and their specific features and notes whether they are offered separately or as part of a system package.

FDC-1 Controller (Digital Systems, 1154 Dunsmuir Place, Livermore CA 94550). The FDC-1 is a microprocessor vased design which will control up to 4 drives. It features write protect, generation and checking of CRC error codes, IBM 3740 format, soft sector formatting, track position verification, and retraction of the head after eight idle revolutions to increase the life of the disk. An especially attractive feature is an automatic bootstrap load from track 0, sector 1 which can be performed at power on *without* host processor intervention.

The commands for the FDC-1 are: File inoperative reset, step the head to another track, step in or step out, enable loading of drive select, read 128 bytes of information into memory, write 128 bytes of information from memory into the previously addressed sector and track.

Control and status information are transferred through an 8-bit input-output port. The information to be read from or written onto the disk is transferred by direct memory access. Prices: FDC-1 controller and cable $520; FDC-1 controller and Shugart SA800 drive $1095; FDC-1 controller and two Shugart SA800 drives $1670. A reference manual package including an FDC-1 manual, a disk drive manual, and diagrams for the Altair bus interface as well as simple 8080 software sells for $5. Note that no interface is provided with this controller but they do provide a schematic for an Altair compatible interface containing about 20 ICs.

GSI FDC300 (General Systems International, Inc., 1440 Allec Street, Anaheim CA 92508).

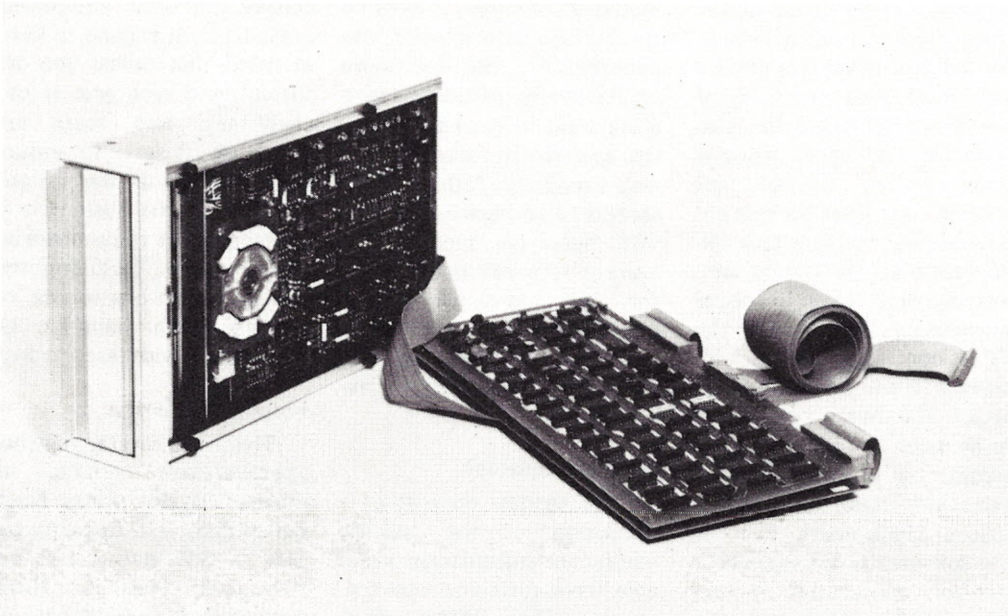
Shugart SA4400 (Shugart Associates, 435 Indio Way, Sunnyvale CA 94086).

SMS FDO300 (Scientific Micro Systems, Inc., 520 Clyde Ave., Mountain View CA 94043).

These three controllers are grouped together since they appear both from the description and photographs to be identical. These units are microprocessor based with fairly high level command information. The eight commands and functions implemented by them are:

RESET: Aborts the current operation and resets controller and all floppy disks in the system.

SEEK: Steps the head to the specified track and verifies correct positioning.



READ: Reads a sector of data from the specified sector.

READID: Reads the next sector ID information.

WRITE: Writes a sector of data with a normal Data Address Marker into the specified sector.

WRDEL: Writes a sector of data with deleted Data Address Marker into the specified sector. Format: Writes address marks, gaps, and data on an entire track according to the IBM format.

STATUS: Returns the status for the addressed drive.

These units include a general purpose host interface which provides 18 TTL signal lines for interface to a host system. This interface requires 3 I/O ports for control, status, and data.

Although all three units appear identical in their specifications and pictures, there are a total of four sets of widely differing dimensions quoted for them with two different sizes listed for the Shugart controller. From the pictures, they appear to be about 6" x 9".

Drive Compatibility:

GSI-FDC300. Not specific except to say . . . IBM compatible drives".

Shugart SA4400. Specifically for the SA400 minifloppy.

SMS FDO300. Orbis 76, Shugart SA800, Calcomp 140, Pertec SD400 and SD500, and Innovex 210. It also says "for IBM compatible drives." Prices: SMS $640, Shugart $490 (for the minifloppy version), GSI not specified.

ICOM CF360 (ICOM Microperipherals, 6741 Variel Ave., Canoga Park CA 91303). Price is $650 for the controller only. (See photo.)

This controller and a Pertec drive are sold together as the "Frugal Floppy" for $1195. Also available are interfaces for the Altair, IMSAI, and Poly 88 for $330 each. The system comes with the FDOS-II software which is described in the software section.

The controller is not microprocessor based and so uses about 125 ICs on two large (7.25" x 15") boards. The commands are: Examine status, Read, Write, Read CRC, Seek, Clear error, flags, Seek track 0, Write DDAM, Load track address, Load unit, Load write buffer, Shift read buffer, Clear, and Examine read buffer. Power requirements are 5 V at 6 Amps and -12 V at 1 Amp.

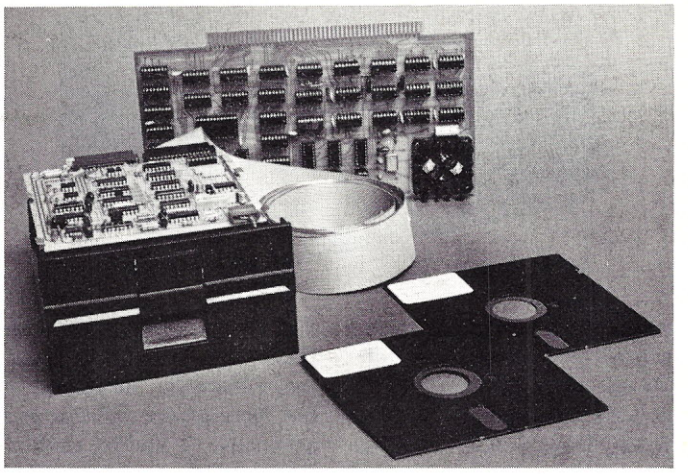
IMSAI Floppy Disk System. The controller has a built-in microprcessor and will handle up to four drives. This is an easy way to get going in a hurry and is priced in kit form at $1449, assembled at $1649, and with two drives at $2374. The disk operating system is available for an additional $40. An extended BASIC with disk access is also supposed to be available.

Floppy Disk 88-DCDD (MITS, 2450 Alamo SE, Albuquerque NM 87106). Price for the system is about $1500. (See photo.)

The controller consists of two boards which plug into the Altair backplane. Up to 16 drives can be addressed by the controller. The disk drive is a Pertec FD400 which is packaged in a 5½" x 17" x 17" box, the same size as the Altair 8800.

This system uses hard sectored disks in a nonstandard format of 32 sectors per track giving a total storage capacity of 310K bytes per disk. Although nonstandard, this format is convenient in that it allows storage of 4K bytes per track, a nice round number which simplifies software housekeeping. But keep in mind the factors discussed in the speed section.

An extended BASIC with disk access is also available for use with this system. This includes programs for copying disks (Mits and copying?), initializing blank disks, etc. A minimum of 20K of RAM is required.



The minifloppy disk system available from North Star Computers which includes the minifloppy disk, controller, Altair bus interface and a disk based BASIC. The disk operating system is on ROM on the controller.

Micro Disk System (North Star Computers, Inc., PO Box 4672, Berkeley CA 94704). Price is $599 for the controller, Altair type interface, and one Shugart Minidisk drive with two disks, one containing the operating system. (See photo.)

The controller is a single Altair compatible board which can control up to three drives. An on board PROM contains the power on bootstrap software. The introductory price includes the controller kit, the minifloppy drive, cables, connectors and two disks. The software included has a file oriented disk operating system and a disk version of an extended BASIC. Additional drives are $425 each.

OSI 470 Floppy Disk System (OSI, 11679 Hayden Street, Hiram OH 44234).

This controller consists of a bare PC board which when populated will plug into the backplane of the OSI 400 computer. It is a very minimal controller which relies on the processor for essentially 100% support. This results in a very inexpensive, yet flexible controller at the expense of speed and the requirement of considerably more software than the intelligent controllers. Price for the PC board, minimal operating system (described as a sector per track system) and a GSI 105 disk is $599.

PerSci 1070 (PerSci, Inc., 4087 Glencoe Ave., Marina Del Rey CA 90291). (See photo.)

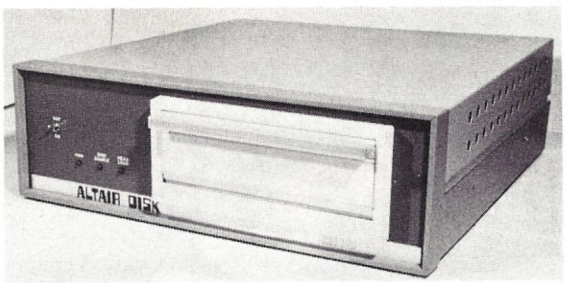
This controller is based on the 8080 microprocessor and as an intelligent controller can communicate by file name and perform all housekeeping functions which would otherwise have to be performed by the host computer. The disk operating software is internal to the controller's 8080 system and includes functions for formatting to the IBM 3740 standard. This unit was designed for use with the PerSci model 70 and 277 drives.

The controller commands are: File, Eject, Initialize, List, Kill (clears and table of contents), Seek Delete, Write, get, and Execute. Writing the necessary operating software for this controller should be very straightforward.

The price is $1195 with the model 70 drive and $1495 with the model 277 dual disk drive.

Sykes OEM Floppy System Kit (Sykes Datatronics, Inc., 375 Orchard St., Rochester NY 14606).

This unit has several noteworthy features. It has hardware address search, automatic sector and track sequencing, a FIFO buffer for asynchronous operation and automatic CRC generation



and detection. In addition the controller electronics are contained on one board which pancakes directly onto the Sykes floppy disk drive to take up a minimum of space. The interface requires only 13 lines — reset, 3 control lines, 1 flag and 8 bidirectional data lines. The price for the controller and one Sykes floppy disk drive is $1398.

TCH Super Simple Interface. This unit designed by *The Computer Hobbyist Magazine* is intended to be a very simple (and therefore cheap) interface to be constructed by the more technically oriented hobbyist. While they are not in the manufacturing business, they do plan to offer a PC board in the Altair format for the controller. Although the board will be in the Altair format it has been designed to be easily adaptable to any other computer. The basic operating system is contained on two 1702A EROMs on the board so the unit is ready to go when the computer is powered up.

The controller is very basic and requires the host computer to do all of the housekeeping duties. It is hard sectored (32 sectors per track) so the data capacity is the same as the Altair disk, 310K bytes. The concession to price that had to be made is average transfer speed. They say that the typical average transfer rate is one sector per revolution, a transfer rate of about 750 bytes per second with the best rate being about 3000 bytes per second. The lower value is lightly slower than a good digital cassette but the access time is still much less. Some small hardware additions should allow higher speed since the bottleneck is the requirement for the host computer to do the parallel-to-serial conversion. The popular 8 bit microprocessors cannot do this at the rate of 250000 bits per second required to keep up with the disk. TCH claims that there will be some spare room on the board for modifications. In addition to being suitable for most, if not all of the home computers, the board has a series of jumpers which allow the controller to be used with almost all of the available drives.

This interface and controller are certainly the cheapest (about $60 for the parts plus the cost of the drive) for the technically oriented hobbyist to get set up.

Software

At present there appear to be only two systems which are being offered separately to the hobbyist. Only one, the Digital Research offering, is specifically intended for the hobbyist to use on a variety of drives and interfaces.

FDOS-II (ICOM, Inc.) $250. This operating system was written by Art Childs, former editor of *SCCS Interface*, and is intended for use with the ICOM FD360 controller. It features variable length named files of any length which may contain program source data, program object data, or user generated data. The commands are:

ASMB — assembles a source file and creates a destination file

BUILD — builds a new file from ASCII input

CHGAT — changes file specifications

COPY — copies disk on unit 0 onto the disk on unit 1

CREAT — creates a file in the directory

DELET — deletes the designated file and repacks the contents of the disk making space for additional files

DUMP — dumps a file to the output device

EDIT — edits a file

HOME — moves the head to track 0

INIT — initializes the file directory

LIST — lists the contents of the file directory

LOAD — loads a file from a reader device into the disk

MERGE — used to concatenate files

MNTR — returns to the host monitor

PRINT — prints the file on the console device

RENAM — renames a file

RUN — loads the file into RAM and begins execution

XGEN — enables system generation of future versions of FDOS-II

This looks like an excellent operating system if you have the ICOM system. It is also not clear whether this is available for the 6800.

CP/M (Digital Research, Box 579, Pacific Grove CA 93950). Price with complete documentation $70. Documentation alone is from $5 to $25.

This system which has been in use for over two years is intended for an 8080 based system (originally the Intel development system). It supports a named file system with up to 64 distinct files on each disk. File storage is dynamically allocated and released with algorithms for optimal read-write head movement. This should help keep the average data transfer rate up by a considerable amount.

The CCP (Command Console Processor) features the basic commands:

DIR — searches the directory

TYPE — outputs a file to the console

REN — renames a file

ERA — erases a file

SAVE — saves memory on the disk for later reload or test

PIP is the peripheral interchange program which controls transfer of files between various devices of the system.

SUBMIT allows the operator to prepare command files with parametric substitution which can then be subsequently executed automatically.

ED is the text editor.

ASM is an assembler which can use either the Intel or Processor Technology assembly language.

DDT is the dynamic debugging tool. This is used for program tracing, debugging and testing.

LOAD loads a file from an Intel hex format file (whatever happened to octal?) ready for execution.

DUMP prints a file in hex onto the console device.

SYSGEN is used to copy a system disk for backup purposes.

While the package was designed for the Intel MDS microcomputer development system it can be easily altered to work with most 8080 based microprocessors with at least 16K of memory and one or more IBM compatible disk drives. The apparently extensive documentation gives instructions for making the patches for other systems.

For hobbyists with IBM format equipment and an 8080 based system, this package looks very hard to beat especially considering the price. What we need to find out is if there is some reasonable way to patch this system into use with the TCH interface.

After seeing the extensive list of goodies that are available, the only two problems that remain are which unit to buy (they all look good), and how to come up with the money. ■

Dennis Brown, President
WAVEMATE
1015 West 190th St.
Gardena CA 90248

The Jupiter II

Once again, we're providing a forum for a manufacturer to discuss his system and point out why he feels you ought to buy it. Last month we gave this spot to Bill Haberhern, President of Infinite, Inc. This month we're turning it over to Dennis Brown, President of Wave Mate for a discussion of the Jupiter II (and in particular, how its bus compares to the Altair bus). — John.

This article, the second in a series on the Wave Mate Jupiter II computer system, discusses some of the design and philosophical decisions that went into the Jupiter II and III CPU cards. It also presents the advantages and disadvantages of some of the well known microprocessor chips. The system bus will be taken into account along with how it can affect the future of a small system. With this increased understanding of the system bus's role, your future decisions about which microcomputers to choose will be made easier.

Which Microprocessor Is Best?

Since every microprocessor has both strong and weak points, a factor in choosing the best one is its intended use. For this article we will assume that the intended use is a small system which can be expanded to the point of becoming a useful small business or development system. Also, since the cost of most microprocessors is less than $100 and the final goal of the system can be one or two orders of magnitude larger, the cost will not be taken into consideration.

We can start by narrowing the field with an important observation; memory is becoming less bulky and more economical at a surprising rate. Therefore, any microprocessor which cannot address at least 64K bytes (65535) is not worthy of consideration because it would severely limit the future economical capability of the system. This eliminates chips such as the Intel 8008, Intel 4004, Intersil 6100, and Signetics 2650.

Another important factor in the selection of the best CPU chip is popularity. If the CPU chip is popular both the manufacturer and users will be able to provide a great deal more software support, which will make your job easier.

These two requirements narrow the field down to four: the Intel 8080, the Motorola 6800, the Zilog Z80, and the MOS Technology 6502, which fall into two distinct groups — the 8080 and Z80 in one and the 6502 and 6800 in the other. The Z80 contains all of the capabilities of the 8080 plus many additional ones, so it is easy to pick the best of these two. The choice is not as distinct between the 6800 and the 6502. The 6502 has more addressing modes than the 6800, but it loses to the 6800 because of the many missing opcodes and branching conditions which make programming easy. The two remaining chips, the Z80 and the 6800, have widely differing architectures. The MC6800 has few internal registers (Fig. 1) but has the ability to manipulate all of memory directly in many ways. The Zilog Z80 has many internal registers (Fig. 2) but no great ability to manipulate memory directly. Since the object of most useful programs is to manipulate data stored in memory, having that capability makes programming in machine language more straightforward.

This gives the edge to the 6800 as far as ease of writing and understanding machine language programs. The Z80, however, has much more powerful loop control instructions where most time is usually spent in a program. This means that program written for the Z80 will run faster.

The choice comes to this: if you want to write only high level programs and the language you want exists on both machines, the Z80 will run faster. However, if you desire to write machine language programs that are easy to write and understand, the 6800 is for you. The choice is yours, but bear in mind that future versions of Z80-type machines will not be significantly easier to understand, while future versions of 6800-type machines will become faster.

What Features Should a CPU Card Have?

The CPU card is the most important card in the system

. . . a father's view

and the most expendable. It is the CPU card that defines the organization and workings of the system bus. It is also the CPU card that is the most prone to obsolescence. New microprocessor chips with increased performance are appearing at a rate exceeding one every six months. For this reason alone the CPU card should not contain any more than the basic function of interfacing the CPU chip to the system bus. That way the least function is discarded when upgrading, or, if more than one type of CPU is desired for the system, the least duplication of circuitry will occur. The most important requirement is to fully support and conform to the bus discipline of the system.

What's a Bus?

Every microcomputer system has a system bus. The bus is a collection of control and data signals that memory and input/output controllers require to connect to the CPU. The system bus is the backbone of your system. Great care should be taken in selecting one that has been designed well with an eye for future expansion.

What Features Should a System Bus Have?

The system bus should be a general type not locked into one microprocessor chip. This way when future microprocessors with enhanced performance features become available, the CPU module can be changed leaving the bulk of the system (memory and device controllers) unmodified. Also, the definitions of the signal lines should be precise and clear so that ambiguities do not arise in future products for a bus. One such example is line 67 of the Altair bus. Two companies have made different products for this bus, one giving line 67 the ability to disable RAM boards, and the other giving it the property of disabling refresh operations.

A general bus is best suited to microprocessors if it has a single, uninverted bidirectional data bus. All micros have a single bidirectional data bus, and future microprocessors will also have a single data bus to save pins on integrated circuit packages. Also, drive capabilities are steadily increasing which gives all the microcomputer systems with a single bus the ability to have microcomputer subsystem chips connect directly to the system bus with the least amount of interfacing.

Other requirements for a bus would be – at least 16 bidirectional address lines, control lines for read/write/I/O and DMA transfers, and eight priority interrupt lines. Of course, not all items have to be implemented to start with, but the capability should be available for future expansion.

A critical requirement for the system bus is that all cards have, next to the bus lines, a heavy ground strip leading directly to the power supply. If on-card regulation is not used, this requirement also is needed for all supply voltages. Without this solid ground, expansion to more than a few cards can lead to reliability problems due to noise sensitivity. One program may run well and another give erroneous results even though it is written correctly.

Likewise, buses using standard TTL drivers should not be expanded to a length greater than 15 inches without placing line terminating resistors at both ends of the bus. Otherwise transmission line effects can cause excessive signal delays and noise sensitivity.

A Word About The "Hobbyist Standard Bus"

Of all the bus systems that have been manufactured one stands out as the volume leader, the Altair bus. This bus has become a sort of defacto standard to the hobbyist.

MITS designed this bus as the basis for an 8080 microprocessor system. The system was successful, which prompted many small companies to design and build add on circuit cards and final-

Fig. 1. Internal registers of the 6800 (the short boxes are 8-bit registers, and the long are 16-bit).

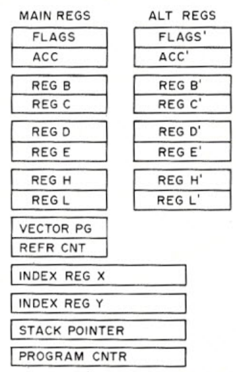


Fig. 2. Internal registers of the Z80.

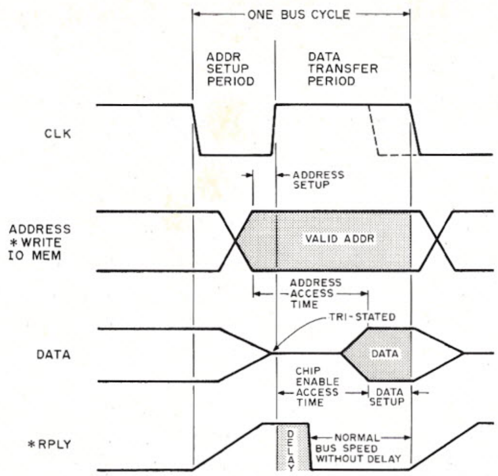


Fig. 3. Jupiter bus timing. By the end of the first portion of the cycle (clock low), the address on the bus will stabilize and all the data drivers will be turned off (tri-stated). By the end of the second portion of the cycle (clock high), the data will be stable. The delay from the start of the second portion of the cycle to the reply signal will be added to the normal bus speed allowing slow devices or memory to control the data transfer time.

ly their own systems, all based on the same bus. While these systems are usable, they lack a bit in elegance. Since, many people own these systems and many fine products exist for them, Wave Mate will be offering an Altair bus extender for the Jupiter system.

The Jupiter Bus: How it Compares To The Altair Bus

The Jupiter system bus consists of 72 pins (for a physical description of the system cards and connectors, see "Why Wire Wrap," *Kilobaud*, January, 1977. The philosophy of this bus was to provide a universal bus for 8-bit microprocessors which, although being as simple and flexible as possible, would allow memory and peripheral controllers to be connected with the least amount of circuitry.

The Altair bus is distinguished most by its 100 signal pins, separate data input and output buses, lack of simple DMA capability, close correlation to the 8080 microprocessor, and widespread use in the hobbyist field.

A system bus breaks down into five basic sections; timing, address, data, interrupt, and direct memory access (DMA). Each section will be discussed separately.

Control/Timing

The basic timing of the Jupiter bus can be seen in Fig. 3. The CLK (clock) line signals that the address is valid and that data transfers may commence. When the clock is off, all the data drivers are turned off, giving a protection band for the previously selected drivers to turn off before the next drivers turn on. Otherwise, large power surges could damage the data drivers.

The *RPLY ($\overline{\text{reply}}$) line is supplied by every device or memory card selected for feedback to the CPU card. Normally a device would complement the CLK signal and put it on the reply line, as in Fig. 4(a). The CPU will measure the delay between the time it sends out the clock signal and the reply signal comes back. This time difference will be added to the basic data transfer time. This means that the bus will run slower or faster depending on how far away the device card is placed from the CPU. If the bus has been buffered to expand the system, the extra delay imposed by the buffers will be added to the basic delay time for those cycles that access memory or controllers on the expanded side of the system. Also, it is very easy for a controller or memory card that cannot run at the full rated bus speed to add an extra delay into the reply line, causing the bus to slow down. See Fig. 4(b).

The *REFR (refresh) line is active for one complete memory cycle. Since no other cycles are active, all the dynamic memories are refreshed at the same time, synchronously with the system. This eliminates any spurious noise caused by a data cycle on the bus and power surges from a dynamic board refreshing asynchronously. Fig. 5 shows the type of circuitry required to connect dynamic RAMs to the Jupiter bus.

The *RESET ($\overline{\text{reset}}$) line will reset the CPU and I/O devices. During power up this line will be on.

The control and timing signals on the Altair bus consist of buffered 8080 inputs and outputs along with a few extras for front panel controls and various clock signals. As such, the bus timing is not very straightforward.

No example is given of the dynamic memory refresh circuitry for the Altair bus because of its complexity. You may have noted the shift from dynamic to static memories for the bus because of this difficulty. Static memories, however, consume much more power, making them undesirable for large main memories.

The Altair bus has provisions for extending the length of its bus cycle. The PRDY line may be made low to cause the processor to wait for a period of time. See Fig. 4(c). However, the Altair bus will wait only in increments of one clock cycle. This is usually one microsecond, making a memory that is slow by 100 nanoseconds wait for a minimum of ten times that long.

Address

The sixteen address lines (AB0-AB15) on the Jupiter are bidirectional tri-state™ lines. These lines can drive a full 10 low power Schottky TTL loads, making it easy to drive all the necessary TTL address selection circuitry.

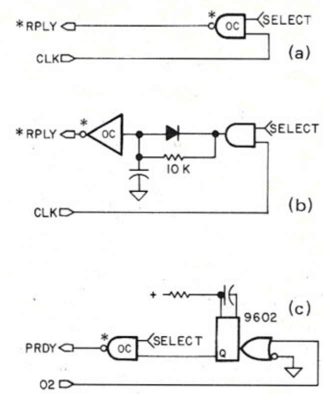


Fig. 4(a). Normal Jupiter bus reply circuit. Every memory or device must generate a reply signal for feedback to the bus timing circuits. (b) Delayed Jupiter reply circuit. Slow devices or memories can delay the reply signal by the amount of the R-C time delay. (c). Slowing an Altair bus cycle. Slow devices or memories on that bus must pull down the ready signal. This method will delay only in increments of the basic clock speed (usually one microsecond).

The eight complemented address decode lines (*A008-*A715) are unique in the flexibility they add to device selection. If one of the 256 I/O devices has been addressed, these lines are the complement of the low order address lines (AB0-AB7). By ANDing the appropriate address lines and address decode lines together, any one of the 256 I/O devices can be selected without inverters. See Fig. 6(b). When not selecting I/O devices, the address decode lines are the complement of the high order address lines (AB8-AB15). By ANDing these lines together, any block of memory down to 256 bytes can be selected, again without inverters.

The *WRITE ($\overline{\text{write}}$) line signals a read or write data transfer is to take place.

The I/O (input/output) line indicates the address on the bus is selecting one of 256 I/O devices and not memory.

The MEM (memory) line indicates the address on the bus is a valid memory address.

The *WP (write protect) line, if on, will cause all preselected memories to become write protected.

The address signals on the Altair bus are made up of sixteen bidirectional address lines, write protect signals of various types and several buffered 8080 outputs. The Altair bus has no complemented address lines, so extra inverters are required to select memory or I/O devices. See Fig. 6.

Data

The eight data lines on the Jupiter bus (DB0-DB7) are bidirectional Tri-state lines. The maximum load allowed on the system is five low-power Schottky TTL loads. This gives one the freedom to add microprocessors and MOS peripheral chips directly to the data bus without external buffer chips on every card. See Fig. 7(b).

The Altair has a separate data-in and data-out bus which requires extra circuitry for every interface. See Fig. 7(a).

Interrupts

Each of the seven interrupt lines on the Jupiter bus (*IRQ0-*IRQ6) will cause the machine to jump to a different interrupt routine. If more than one line is pulsed at a time, the highest number will prevail. The CPU has the ability to turn off interrupts on all levels below any selected level. The eighth interrupt level goes directly to the front panel interrupt switch.

The *VCTO (vector out) and *VCTI (vector in) lines are daisy-chained down the bus so that the *VCTO line of one module connects to the *VCTI line of the next module. The signal originates at the CPU module and is passed from one module to the next. This signal indicates that the CPU is processing an interrupt and is requesting an interrupt vector from the memory. This allows an I/O device or system monitor to relocate a CPU's dedicated vector areas to someplace that is more versatile for system use.

The Altair bus has provisions for eight vectored interrupts, although not implemented for the bus on most CPU cards. The 8080 interrupt request, enable, and acknowledge also appear on the bus and these are what are usually used.

Direct Memory Access (DMA)

The *HALT (halt) line is used by a DMA device to halt the processor so the maximum DMA data rate on the bus can be achieved.

The *DREQ (DMA request) line is used by a DMA device with noncritical data requirements to cause a maximum of one DMA cycle per instruction executed.

The *DMAO (DMA out) and *DMAI (DMA in) lines are also daisy-chained down the bus so that the *DMAO line of one module connects to the *DMAI line of the next module. The signal originates at the CPU module and is passed from one module to the next. This signal indicates that a DMA device may have the current cycle for a DMA cycle. A DMA device must stop the propagation of this signal if it is going to do a DMA transfer (Fig. 8).

The *ENA (enable DMA address) line has a dual function. First, during the address setup time (CLK off) a DMA device will signal with this line that an address has been placed on the address bus and a valid DMA transfer is to take place. Second, during the data transfer time (CLK on), this line will signal the clock circuitry to wait for this signal instead of for the *RPLY line. This allows a DMA device which cannot run at the full memory speed to slow down the basic cycle time of the CPU.

The Altair bus has very limited DMA capability. Only a single DMA device can be plugged into the bus at any one time. The DMA device

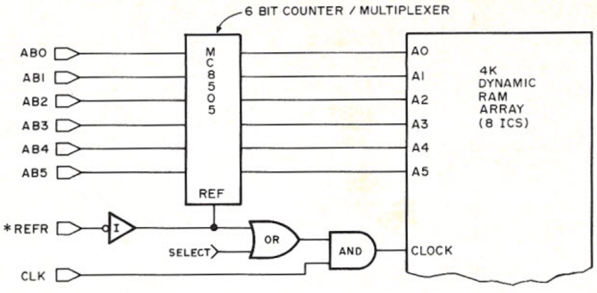


Fig. 5. Refreshing dynamic memories from the Jupiter bus. All dynamic memories are refreshed simultaneously on the Jupiter bus when the refresh signal appears on the bus. No other cycles are going on during this period. The MC8505 is a six-bit counter/multiplexer that substitutes the counter outputs for the address inputs during a refresh operation.

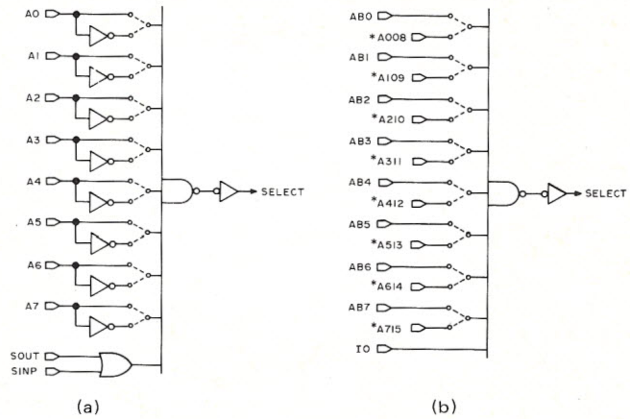


Fig. 6. Device selection circuitry. (a) Selecting one of 256 I/O bytes on the Altair bus requires inverters on every address line and ORing of the INPUT and OUTPUT status lines. (b) Selecting one of 256 I/O bytes on the Jupiter bus does not require any inverters on the address lines.

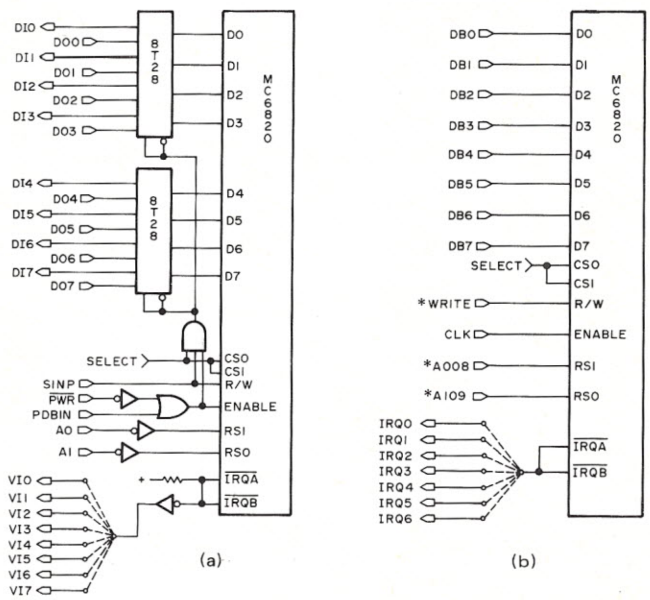


Fig. 7. Connecting a parallel interface to two buses. (a) The Altair bus requires many external chips to interface the data and control lines. Total conservative count for a parallel interface is seven integrated circuits. (b) The Jupiter requires no external chips to interface the straightforward data and control lines. Total count for a Jupiter parallel interface is two and one-half integrated circuits.

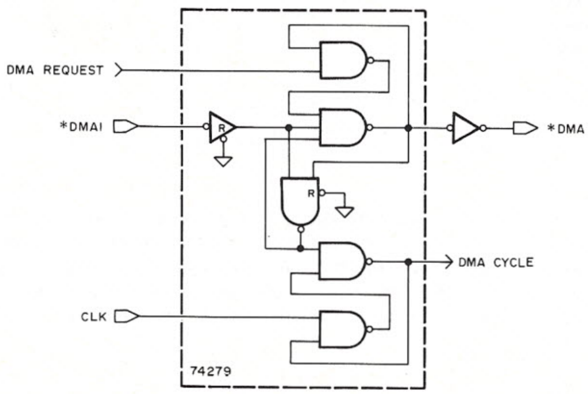


Fig. 8. Circuit for connecting multiple DMA devices to the Jupiter bus priority chain. The circuit connects between the DMA input line and the DMA output line. The DMA signal is not passed to the next devices further down the bus if this device is requesting a DMA cycle. This entire circuit can be built using a single 74279 quad set-reset flip-flop.

must first halt the processor with the PHOLD then wait for the PALDA signal. Then the address, data output, command/control and status buffers must be disabled.

The DMA device must then supply all of the above information to the bus in sync with the basic 8080 clock. The timing of this circuitry takes a lot of logic and could easily fill an entire card. For example, look at the Cromemco Dazzler cards – one is just for DMA.

Putting It All Together

The circuitry of Figs. 4, 6, and 7 constitute a complete 16-bit parallel interface, one for the Jupiter bus (4b, 6a, 7b), and the other for the Altair bus (6a, 7a). The one for the Jupiter bus takes 2½ ICs, while the one for the Altair takes 7 ICs. It is no trivial task to get on and off the Altair bus.

The job of putting other microprocessors onto the Altair bus is also not simple. The problem is that the bus was designed too closely around the 8080 chip. Other microprocessors do not have the same signals as the 8080 to put onto the bus, so many signals are either ignored or kludged. This causes a shift in the meanings of all the signals as each new generation of microprocessors is introduced to the bus. I think it is about time for the Altair bus to get its act together and redefine its signals more generally before it's too late.

In the meantime the Jupiter bus now has two CPU cards available for it. Neither has changed the basic signal timing or definitions even though the CPU chips are quite different.

The Jupiter II and III CPU Cards

The Jupiter II MC6800 CPU card and the Jupiter III Z80 CPU card are presented together to demonstrate how two different microprocessor chips can be designed for the same bus system. (For a photo of Jupiter II CPU card refer to "Why Wire Wrap" *Kilobaud*, January, 1977.) These are only two of the many possible CPU cards the Jupiter series can support. There will be many new

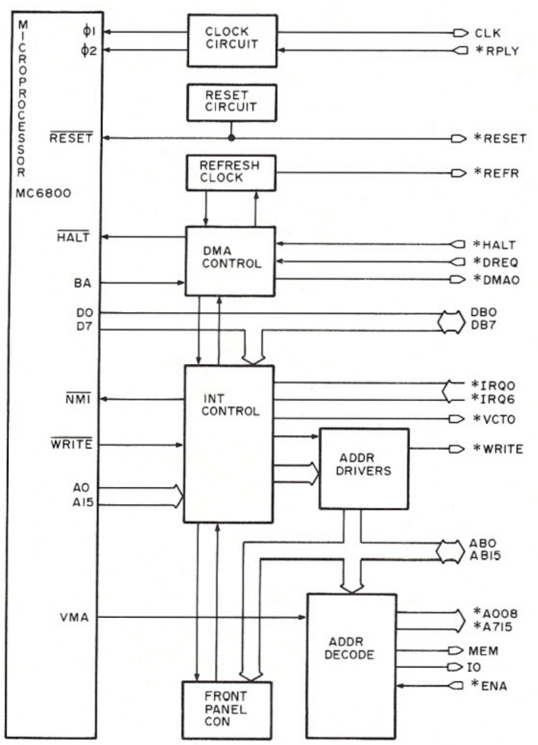


Fig. 9. Block diagram of the functional units required in the Jupiter II 6800 CPU card to interface to the Jupiter bus.

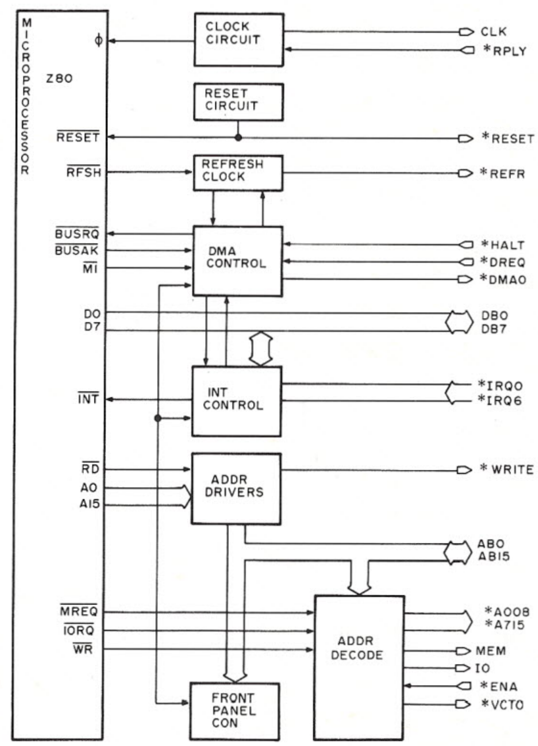


Fig. 10. Block diagram of the functional units required in the Jupiter III Z80 CPU card to interface to the Jupiter bus.

microprocessors adapted to this bus in the future. Refer to Figs. 9 and 10 for the block diagrams of the two CPU cards as each of these functional units will be discussed in turn.

Microprocessor. The microprocessor in both cases forms the heart of the circuit. In the Jupiter II the microprocessor is the Motorola MC6800. In the Jupiter III the microprocessor is the Zilog Z80.

Clock circuit. The clock circuit provides the basic timing signals for the microprocessor and the system bus. Both the clock and the *RPLY line from the bus interact as described in the bus section. This circuit also generates the master clock for the microprocessor.

Refresh clock. This circuit times out the refresh signal on the bus, then takes the next available refresh time from the microprocessor and gives the signal for all the dynamic memories in the system to refresh. For the 6800 this causes a 7% decrease in processor throughout while the Z80 pays no time penalty. This trade-off for the 6800 was made to simplify the memory system and make it more reliable.

Reset generator. This circuit provides the power on reset pulse to the entire system.

DMA control. This circuit samples the two bus DMA requests and generates the *DMAO signal after halting the microprocessor.

Interrupt control. Eight priority interrupts are sampled by this logic (the eighth level not appearing on the bus goes to the front panel interrupt switch). The *VCTO signal is generated in response to an interrupt. The CPU can write a status word to this logic and disable all levels lower than a selected one.

Address line drivers. The address lines from the microprocessor are buffered by this circuit from the five LSTTL loads usually driven by the microprocessor to 10 LSTTL loads required by the bus.

Address decode. This circuit detects which kind of bus transfer is to take place and generates the MEM and I/O signals. During a DMA cycle, the MEM and I/O signals will not be generated until the *ENA (enable address) line is turned on by the DMA device. This circuit also transfers the correct half of the complemented address bus to the eight address decode lines for an I/O transfer.

Data lines. The data lines are driven directly by the microprocessor.

Front panel connector. The front panel for the system connects to a connector on the back of the CPU card instead of to the bus. This keeps the front panel from using an extra bus slot and allows it to control signals inside the CPU that would not normally be required on a general purpose bus for memory or device controllers.

Conclusion

The various microprocessors have been examined for their suitability for a microcomputer system. One clear cut leader was found (Z80) but straightforward programming was lacking. By the time this article goes to print, we may very well have a new contender for a high performance microprocessor. The conclusion is that the processor is not as important as having a system flexible enough to support future microprocessors. The requirements for a good general purpose microcomputer bus were also discussed, with specific recommendations for flexibility and reliability. The Jupiter bus was compared to the Altair bus and many shortcomings were found in the latter. An example of two CPU cards, one built around the MC6800 and one around the Z80, was shown to exemplify the flexibility you can have with a microcomputer system built around a general purpose bus. ■

Gordon W. Flemming
13490 Simshaw Avenue
Sylmar CA 91342

How to Win $25,000 of Your Own Money

... Keno game program

```
RUN

               $$$$$ K E N O $$$$$
                   $$$$$$$$$

DO YOU WANT DIRECTIONS? YES

   EACH PLAY WILL COST YOU $1. YOU MAY SELECT
FROM 1 TO 15 NUMBERS TO PLAY. WHEN ASKED FOR
YOUR NUMBERS, TYPE IN EACH NUMBER SEPARATED
BY COMMAS. IF YOU ACCIDENTLY PLACE A COMMA
AFTER YOUR LAST NUMBER, THE COMPUTER WILL
TELL YOU THAT THE EXTRA NUMBER WAS IGNORED (IN
THIS CASE, THE EXTRA WAS A SPACE). IF YOU PUT IN
MORE NUMBERS THAN NEEDED, ONLY THE FIRST RE-
QUIRED ONES WILL BE USED. IF YOU DON'T ENTER
ENOUGH NUMBERS, THE COMPUTER WILL RESPOND
WITH '??'. ENTER THE REQUIRED EXTRA NUMBERS.

   THE COMPUTER WILL BE CHOOSING 20 NUMBERS AT
RANDOM FROM NUMBER 1 TO NUMBER 80.

TO QUIT, TYPE '0' WHEN ASKED HOW MANY NUMBERS YOU WANT.

G O O D   L U C K ! ! ! !


IS THIS A CONTINUATION OF AN OLD GAME? NO
HOW MUCH CASH DID YOU BRING? 5

HOW MANY NUMBERS DO YOU WANT? 1
YOUR NUMBERS:
? 73
MY NUMBERS WERE:

   37 50 73 35 62 10 15 51 64 78
   47 20 44 1 23 12 49 8 48 74

YOU HAD 1 RIGHT. YOU NOW HAVE 7 DOLLARS LEFT.

HOW MANY NUMBERS DO YOU WANT? 2
YOUR NUMBERS:
? 73,37
MY NUMBERS WERE:

   19 63 75 10 16 65 50 36 38 67
   8 70 42 5 35 54 6 14 26 13

YOU HAD 0 RIGHT. YOU NOW HAVE 6 DOLLARS LEFT.

HOW MANY NUMBERS DO YOU WANT? 10
YOUR NUMBERS:
FIRST FIVE:? 11,22,33,44,55
LAST FIVE:? 73,37,1,80,40
MY NUMBERS WERE:

   23 75 59 56 35 66 9 3 58 29
   62 2 61 20 50 26 5 53 16 38

YOU HAD 0 RIGHT. YOU NOW HAVE 5 DOLLARS LEFT.

HOW MANY NUMBERS DO YOU WNAT? 1
YOUR NUMBERS:
? 73
MY NUMBERS WERE:

   35 5 63 46 75 6 23 66 64 38
   3 80 70 13 2 39 67 73 20 36

YOU HAD 1 RIGHT. YOU NOW HAVE 7 DOLLARS LEFT.

HOW MANY NUMBERS DO YOU WANT? 10
YOUR NUMBERS:
FIRST FIVE:? 70,71,71,73,74
LAST FIVE:? 75,76,77,78,79
YOU HAVE DUPLICATED NUMBERS. TRY AGAIN.
YOUR NUMBERS:
FIRST FIVE:? 70,71,72,73,74
LAST FIVE:? 75,76,77,78,79
MY NUMBERS WERE:

   76 4 77 13 59 29 35 36 55 49
   32 33 67 10 78 54 57 38 28 27

YOU HAD 3 RIGHT. YOU NOW HAVE 6 DOLLARS LEFT.

HOW MANY NUMBERS DO YOU WANT? 0
COME BACK AGAIN. BRING MORE MONEY!!!
TOTAL TURNS 5
TOTAL MONEY LEFT 6

OK
```

The following program is a true simulation of the game of Keno as played in Las Vegas. I started writing the program after finding that I couldn't remember enough probability and statistics to figure what the payoffs should be to give even-money odds. One thing led to another and this program was written to actually play the game.

Keno is very popular because there is the possibility of a huge payoff ($25,000) for a small wager ($1). "Keno runners" go around the casinos gathering Keno cards from the players. The Keno cards look somewhat like Bingo cards and the player circles the numbers he wishes to play.

The player selects from 1 to 15 numbers to play. Twenty numbers are then drawn at random from the number 1 to 80. The payoff varies with how many numbers are correct. For instance, 3 correct out of 3 guesses pays $43.00 but 3 correct out of 7 guesses pays $0.50.

In the casinos, new games are run at frequent intervals. The results are displayed on "Keno boards" throughout the casino.

The following program can be simplified if one wants to play a fixed quantity of guesses. For instance, if you want to play 7 numbers, delete lines 500 through 800 and lines 850 through 1210. The input routine in lines 190 through 226 can also be modified by letting T = 7 and deleting lines 200 through 205 and 207 through 226.

Using seven numbers the payoff would be:

Number Correct	*Payoff*
3	$ .50
4	1.50
5	21.50
6	328.50
7	5000.00

Additional savings in memory used can be made by deleting the remark statements. Also consider using multiple statements per line. Check first to be sure no GOTO or GOSUB statements are directed to the second or third statements of a multiple statement line. The listing shown has remarks and single statement lines for easier understanding of the program operation.

Upon termination, the program will print the total number of plays and the amount of money left. This is to allow continuation of the game. The real game takes about one-half hour. So you can figure your cost per hour for the entertainment. (It's cheaper than slot machines but not nearly as cheap as a well-played Blackjack game.)

After entering the program, settle into an easy chair and talk your wife, child, or girl friend into being your "Keno runner." You might even order a drink once-in-a-while, but tip big. Settle back and figure out your own "system" to beat the odds. ■

```
LIST

10 REM KENO ACCORDING TO PAYOFFS OF
11 REM MGM GRAND CASINO LAS VEGAS JAN 1976
12 REM WRITTEN BY G.W.FLEMMING MEMBER SCCS
15 PRINTTAB(15);"$$$$$ K E N O $$$$$"
16 PRINTCHR$(7);CHR$(7);CHR$(7);TAB(15);"     $$$$$$$$$";CHR$(7)
17 PRINTCHR$(7);CHR$(7);CHR$(7);CHR$(7);CHR$(7)
18 INPUT"DO YOU WANT DIRECTIONS";D$
19 IF LEFT$(D$,1)="N" THEN 145
20 PRINT:PRINT" EACH PLAY WILL COST YOU $1. YOU MAY SELECT"
21 PRINT"FROM 1 TO 15 NUMBERS TO PLAY. WHEN ASKED FOR YOUR"
22 PRINT"NUMBERS, TYPE IN EACH NUMBER SEPARATED BY COMMAS."
23 PRINT"IF YOU ACCIDENTLY PLACE A COMMA AFTER YOUR LAST"
24 PRINT"NUMBER, THE COMPUTER WILL TELL YOU THAT THE EXTRA"
25 PRINT"NUMBER WAS IGNORED ( IN THIS CASE, THE EXTRA WAS"
26 PRINT"A SPACE). IF YOU PUT IN MORE NUMBERS THAN NEEDED,"
27 PRINT"ONLY THE FIRST REQUIRED ONES WILL BE USED. IF"
28 PRINT"YOU DON'T ENTER ENOUGH NUMBERS, THE COMPUTER WILL"
29 PRINT"RESPOND WITH '??'. ENTER THE REQUIRED EXTRA NUMBERS."
30 PRINT:PRINT"          THE COMPUTER WILL BE CHOOSING 20 NUMBERS"
31 PRINT"AT RANDOM FROM NUMBER 1 TO NUMBER 80."
32 PRINT:PRINT"TO QUIT, TYPE '0' WHEN ASKED HOW MANY NUMBERS YOU WANT."
33 PRINT:PRINT:PRINT"G O O D   L U C K ! ! ! !"
34 PRINT:PRINT
145 DIM A(20): DIM H(80)
146 INPUT"IS THIS A CONTINUATION OF AN OLD GAME";A$
147 IF LEFT$(A$,1)="N" THEN 151
148 INPUT"HOW MANY PREVIOUS TURNS";E
149 INPUT"HOW MANY DOLLARS WERE LEFT";S
150 GOTO 160
151 E=0
155 INPUT"HOW MUCH CASH DID YOU BRING";S
160 W=S
170 Q=0:PRINT
180 REM INPUT NUMBERS
190 INPUT"HOW MANY NUMBERS DO YOU WANT";T
191 IF T=0 THEN 2000
195 PRINT"YOUR NUMBERS:"
200 IF T=1 THEN INPUT A(1)
201 IF T=2 THEN INPUT A(1),A(2)
202 IF T=3 THEN INPUT A(1),A(2),A(3)
203 IF T=4 THEN INPUT A(1),A(2),A(3),A(4)
204 IF T=5 THEN INPUT A(1),A(2),A(3),A(4),A(5)
205 IF T=6 THEN INPUT A(1),A(2),A(3),A(4),A(5),A(6)
206 IF T=7 THEN INPUT A(1),A(2),A(3),A(4),A(5),A(6),A(7)
207 IF T=8 THEN INPUT A(1),A(2),A(3),A(4),A(5),A(6),A(7),A(8)
208 IF T=9 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
209 IF T=9 THEN INPUT"LAST FOUR:";A(6),A(7),A(8),A(9)
212 IF T=10 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
213 IF T=10 THEN INPUT"LAST FIVE:";A(6),A(7),A(8),A(9),A(10)
214 IF T=11 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
215 IF T=11 THEN INPUT"LAST SIX:";A(6),A(7),A(8),A(9),A(10),A(11)
216 IF T=12 THEN INPUT"FIRST SIX:";A(1),A(2),A(3),A(4),A(5),A(6)
217 IF T=12 THEN INPUT"LAST SIX:";A(7),A(8),A(9),A(10),A(11),A(12)
218 IF T=13 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
219 IF T=13 THEN INPUT"NEXT FIVE:";A(6),A(7),A(8),A(9),A(10)
220 IF T=13 THEN INPUT"LAST THREE:";A(11),A(12),A(13)
221 IF T=14 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
222 IF T=14 THEN INPUT"NEXT FIVE:";A(6),A(7),A(8),A(9),A(10)
223 IF T=14 THEN INPUT"LAST FOUR:";A(11),A(12),A(13),A(14)
224 IF T=15 THEN INPUT"FIRST FIVE:";A(1),A(2),A(3),A(4),A(5)
225 IF T=15 THEN INPUT"NEXT FIVE:";A(6),A(7),A(8),A(9),A(10)
226 IF T=15 THEN INPUT"LAST FIVE:";A(11),A(12),A(13),A(14),A(15)
227 FOR R=1 TO (T-1)
230 FOR M=R+1 TO T
235 IF A(R)=A(M) THEN 255
236 IF A(R)>80 THEN 261
240 NEXT M
245 NEXT R
250 GOTO 299
255 PRINT"YOU HAVE DUPLICATED NUMBERS. TRY AGAIN."
260 GOTO 191
261 PRINT"YOU HAVE A NUMBER TOO LARGE. TRY AGAIN."
262 GOTO 191
299 N=T
300 FOR X=1 TO 20:REM SELECT 20 NUMBERS AT RANDOM
310 H(X)=INT(80*RND(8))+1
```

```
320 FOR Y=1 TO(X-1)
330 IF H(X)=H(X-Y) THEN 310
340 NEXT Y
350 NEXT X
399 REM DETERMINE HOW MANY WINNERS
400 FOR X=1 TO N
410 FOR Y=1 TO 20
420 IF A(X)=H(Y) THEN Q=Q+1
430 NEXT Y
440 NEXT X
499 REM DETERMINE PAYOFF
500 IF N<>1 THEN 550
510 IF Q=1 THEN W=W+3
550 IF N<>2 THEN 600
560 IF Q=2 THEN W=W+12
600 IF N<>3 THEN 650
610 IF Q=2 THEN W=W+1
620 IF Q=3 THEN W=W+43
650 IF N<>4 THEN 700
660 IF Q=2 THEN W=W+1
670 IF Q=3 THEN W=W+4
680 IF Q=4 THEN W=W+114
700 IF N<>5 THEN 750
710 IF Q=3 THEN W=W+1.5
715 IF Q=4 THEN W=W+21.5
720 IF Q=5 THEN W=W+485.5
750 IF N<>6 THEN 800
760 IF Q=3 THEN W=W+1
765 IF Q=4 THEN W=W+4.5
770 IF Q=5 THEN W=W+85.5
775 IF Q=6 THEN W=W+1571.5
800 IF N<>7 THEN 850
810 IF Q=3 THEN W=W+.5
815 IF Q=4 THEN W=W+1.5
820 IF Q=5 THEN W=W+21.5
825 IF Q=6 THEN W=W+328.5
830 IF Q=7 THEN W=W+5000
850 IF N<>8 THEN 900
855 IF Q=5 THEN W=W+8.5
860 IF Q=6 THEN W=W+85.5
865 IF Q=7 THEN W=W+1643
870 IF Q=8 THEN W=W+17857
900 IF N<>9 THEN 950
905 IF Q=4 THEN W=W+.5
910 IF Q=5 THEN W=W+3
920 IF Q=6 THEN W=W+43
925 IF Q=7 THEN W=W+285.5
930 IF Q=8 THEN W=W+4000
935 IF Q=9 THEN W=W+17857
950 IF N<>10 THEN 1000
955 IF Q=5 THEN W=W+2
956 IF Q=6 THEN W=W+20
957 IF Q=7 THEN W=W+140
958 IF Q=8 THEN W=W+1000
959 IF Q=9 THEN W=W+3800
960 IF Q=10 THEN W=W+17857
1000 IF N<>11 THEN 1050
1001 IF Q=5 THEN W=W+1
1002 IF Q=6 THEN W=W+8.5
1003 IF Q=7 THEN W=W+71.5
1004 IF Q=8 THEN W=W+357
1005 IF Q=9 THEN W=W+1714
1006 IF Q=10 THEN W=W+10714
1007 IF Q=11 THEN W=W+17857
1050 IF N<>12 THEN 1100
1055 IF Q=5 THEN W=W+1
1056 IF Q=6 THEN W=W+4
1057 IF Q=7 THEN W=W+28.5
1058 IF Q=8 THEN W=W+214
1059 IF Q=9 THEN W=W+571.5
1060 IF Q=10 THEN W=W+1428.5
1061 IF Q=11 THEN W=W+7143
1062 IF Q=12 THEN W=W+25000
1100 IF N<>13 THEN 1150
1101 IF Q=6 THEN W=W+1.5
1102 IF Q=7 THEN W=W+17
1103 IF Q=8 THEN W=W+71.5
1104 IF Q=9 THEN W=W+678.5
1105 IF Q=10 THEN W=W+3571.5
1106 IF Q=11 THEN W=W+6428.5
1107 IF Q=12 THEN W=W+14285.7
1108 IF Q=13 THEN W=W+25000
1150 IF N<>14 THEN 1200
1160 IF Q=6 THEN W=W+3
1161 IF Q=7 THEN W=W+8
1162 IF Q=8 THEN W=W+31.5
1163 IF Q=9 THEN W=W+250
1164 IF Q=10 THEN W=W+714
1165 IF Q=11 THEN W=W+2857
1166 IF Q=12 THEN W=W+10714
1167 IF Q=13 THEN W=W+21428.5
1168 IF Q=14 THEN W=W+25000
1200 IF N<>15 THEN 1900
1201 IF Q=6 THEN W=W+1.5
1202 IF Q=7 THEN W=W+7
1203 IF Q=8 THEN W=W+21.5
1204 IF Q=9 THEN W=W+107
1205 IF Q=10 THEN W=W+285.5
1206 IF Q=11 THEN W=W+2143
1207 IF Q=12 THEN W=W+7143
1208 IF Q=13 THEN W=W+21428.5
1209 IF Q=14 THEN W=W+25000
1210 IF Q=15 THEN W=W+25000
1899 REM PRINT GAME RESULTS
1900 PRINT"MY NUMBERS WERE:"
1905 PRINT
1910 FOR X=1 TO 10
1920 PRINTH(X);
1930 NEXT X
1935 PRINT
1940 FOR X=11 TO 20
1950 PRINTH(X);
1960 NEXT X
1965 PRINT:PRINT:W=W-1:REM SUBTRACT BET
1966 E=E+1
1967 IF W<1 AND W>0 THEN PRINT"YOU HAD";Q;"RIGHT,BUT NOW YOU"
1968 IF W<1 AND W>0 THEN PRINT"DON'T HAVE ENOUGH TO BET."
1969 IF W<1 AND W>0 THEN GOTO 2000
1970 IF W=0 THEN PRINT"YOU HAD";Q;"RIGHT, BUT NOW YOU'RE BROKE."
1971 IF W=0 THEN 2000
1972 PRINT"YOU HAD";Q;"RIGHT. YOU NOW HAVE";W;"DOLLARS LEFT."
1980 GOTO 170
2000 PRINT"COME BACK AGAIN. BRING MORE MONEY!!!"
2010 PRINT"TOTAL TURNS";E
2020 PRINT"TOTAL MONEY LEFT";W
2500 END
OK
```

LIKE WAVE MATE

Now Wave Mate puts them together for you — the Jupiter system, the new high performance dual floppy disk drive from PerSci and new flexible software.

Imagine what you can do with a disk drive that seeks over five times faster than the closest competitor.

Imagine what you can do with a computer system that's wire-wrapped so it can be upgraded with advancements in technology.

Imagine what you can do with a series of high level compilers so flexible that the software you write for today's hot microprocessor will run on tomorrow's.

No one but Wave Mate can put a flexible package like this together for you.
Can you imagine any reason why you should settle for less? We can! You can start smaller with the Jupiter A system without sacrificing the quality and future growth capability of your computer system and you have your choice of 6800 or Z80 processors.

Send information on:
- ☐ Floppy disk system
- ☐ Jupiter IIA system
- ☐ Jupiter IIIA system

NAME ____________________

ADDRESS ____________________

CITY __________ STATE ________ ZIP ________

Wave Mate

WAVE MATE 1015 West 190th Street, Gardena, California 90248 Telephone (213) 329-8941
Dept. 22

Using the "$50" Terminal

I swore up and down that if I ever saw another article dealing with ASCII-to-Baudot and Baudot-to-ASCII conversion, I was going to throw up my hands and scream. (Which is exactly what this levelheaded individual did when he received the following article!) But . . . my reaction changed after reading it. I don't care how many articles, in how many other magazines, have preceded it; Jim Brown has got a winner here. The reasons why are numerous. First of all, there is no denying the fact that there are a variety of Baudot machines out there which perform very nicely as terminals for the home computer (and the machine itself is usually less expensive than most of the printers on the market). Jim's article doesn't include listings of the program he wrote . . . but instead, he provides a complete set of flowcharts so the somewhat-higher-than-novice programmer can develop the routines for any home system. He's got some good definitions for the terms he uses, and it's all there . . . the software and the hardware. Try it, you'll like it. — John.

James Brown
13865 Mira Montana
Del Mar CA 92014

After spending nearly two years of effort designing, building, and debugging my own scratch built 16-bit minicomputer, I came to the hard realization that to be usable during development of any software, my machine simply had to be equipped to produce hard copy. After I shopped around for a used 7 level ASCII Teletype, I realized that one would cost at least $600.00 and then be in questionable condition.

During my search, I ran into several 5 level Baudot machines that could be picked up for a song. Whenever I found one of this type, I bought it and stored it in my garage while I went on with the great ASCII TTY hunt.

When I finally gave up the search and started considering the pile of Baudot machines, several problems appeared: 1) There are more keys on an ASCII machine than my Baudot clunkers, and if I ever wanted to talk to software that spoke only ASCII, I would be short of conversation because of the different number of keys. 2) A standard ASR33 Teletype runs at 110 baud, where my machines run at either 45.5 or 75 baud. This meant that if I wanted to write a Teletype look-alike program and hook my computer to the telephone lines to converse with the local time-share service, I would have a speed problem. 3) I hadn't the slightest idea how to interface my computer with a teletypewriter, regardless of the type.

I solved these problems and went on to bigger and better things. I wrote a cross assembler that runs on the time-share service, is accessed by my computer, and downloads the assembled object through the telephone lines. I am now working on a real time executive to make all my peripherals play together (a high speed paper tape punch and optical reader, twin cassettes, two more TTYs, my ham rig, and a modem).

When I realized that the solution to my problems could also be a key to other computer builders who have a microprocessor but can't afford an ASR33, I felt that it should be written up. The remainder of this article describes my solutions to three basic problems:

Solution 1. Code conversion — software converter that makes use of the Baudot keyboard to produce all ASCII codes and printer to print all ASCII characters.

Solution 2. Speed differential — software ring buffer and a working definition of a baud.

Solution 3. Computer/TTY interface — hardware design that includes a UART, optical isolators, and loop supply.

Code Conversion

This problem boils down to making my computer look like an ASR33 type terminal. To do this, my computer had to:

- Accept 7-bit ASCII characters from the driving software, code convert and output to a Baudot printer.
- Where no equivalent Baudot character exists, output a two-letter equivalent, e.g., .EQ. for " = ".
- For ASCII characters that are normally nonprinting, such as SOM, output a "?" and continue.
- Accept Baudot keyboard characters, code convert and output the ASCII equivalent.
- Accept two character Baudot symbols for non-Baudot characters (.GT. for the ASCII " > ").
- Allow control characters to be formed and output.

What's a Driver

This conversion software is contained in a Teletype driver. Drivers are the things that keep a system programmer from going crazy trying to take care of all the idosyncrasies of each device he is talking to. They accept data to output and do all the code conversion, parity testing, and keeping track of the system status. When you hit a key on my Teletype, it is the driver that feels it first and does things like shift from letters to figures and back, keeps track of which was done last, and converts the four symbols ".GT." to the single ASCII character for *greater than.* It is the driver that frees the using software from all these detail chores.

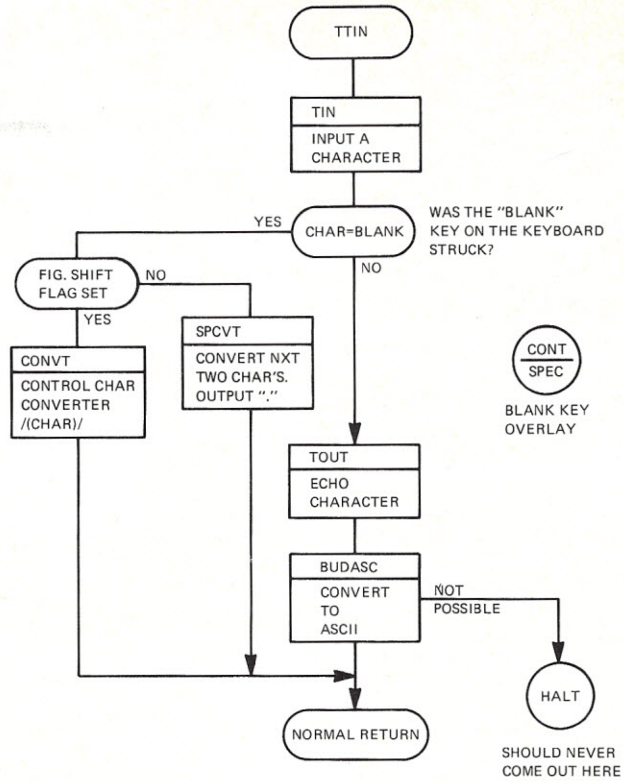


Fig. 1. TTY Input Routine (TTIN).

The code chosen to work between the using software and the driver is 8-bit ASCII with no parity bit. Any code could have been used; but since ASCII is a standard that most existing software expects, it was chosen.

The driver consists of an input and an output side; but since some subroutines are shared between them, you can only use one side at a time. If you want a full duplex software system, you could duplicate the shared subroutines or write them to be reentrant. If you haven't run across this term before, it means *a subroutine that is capable of being entered by one calling routine and then before completion having an interrupt occur that causes the same routine to be re-entered by a second calling routine.* Reentrancy is a very nice feature, but rather complex for this application.

Since your computer is an order of magnitude faster than a printer or keyboard, the driver can easily run through the input side when you hit a key, then lock out the keyboard while it echoes the character on the printer.

Input

The subroutine TTIN (see Fig. 1) is used to input from the keyboard. Its output will be a single ASCII character ready for the using software. TTIN calls the subroutine TIN (Type IN) to get the Baudot character. It then examines it to see if the *blank* key on the keyboard was stuck. It is this key, the only extra one on a Baudot keyboard, that makes the conversion possible. I have glued a legend on this key which has

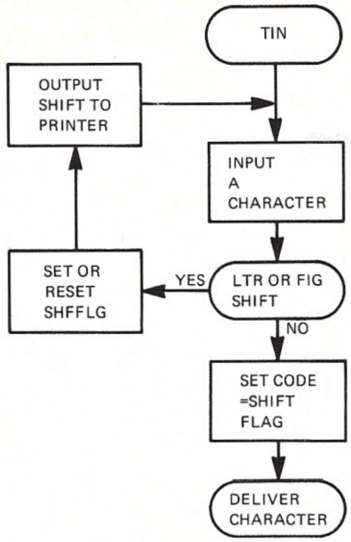


Fig. 2. Input a Character (TIN).

	Word in Table			Word in Table	
CHAR	BAUDOT	ASCII	CHAR	BAUDOT	ASCII
A	030	101	1	235	61
B	023	102	2	231	62
C	016	103	3	220	63
D	022	104	4	212	64
E	020	105	5	201	65
F	026	106	6	225	66
G	013	107	7	234	67
H	005	110	8	214	70
I	014	111	9	203	71
J	032	112	0	215	60
K	036	113	(Space)	204	040
L	011	114	!	226	041
M	007	115	"	221	042
N	006	116	$	222	044
O	003	117	&	213	046
P	015	120	(	236	050
Q	035	121	)	211	051
R	012	122	'	206	054
S	024	123	-	230	055
T	001	124	.	207	056
U	034	125	/	227	057
V	017	126	:	216	072
W	031	127	;	217	073
X	027	130	?	223	077
Y	025	131	#	205	043
Z	021	132	C/R	002	215
			L/F	010	212

Fig. 4. ASCII/Baudot Conversion Table.

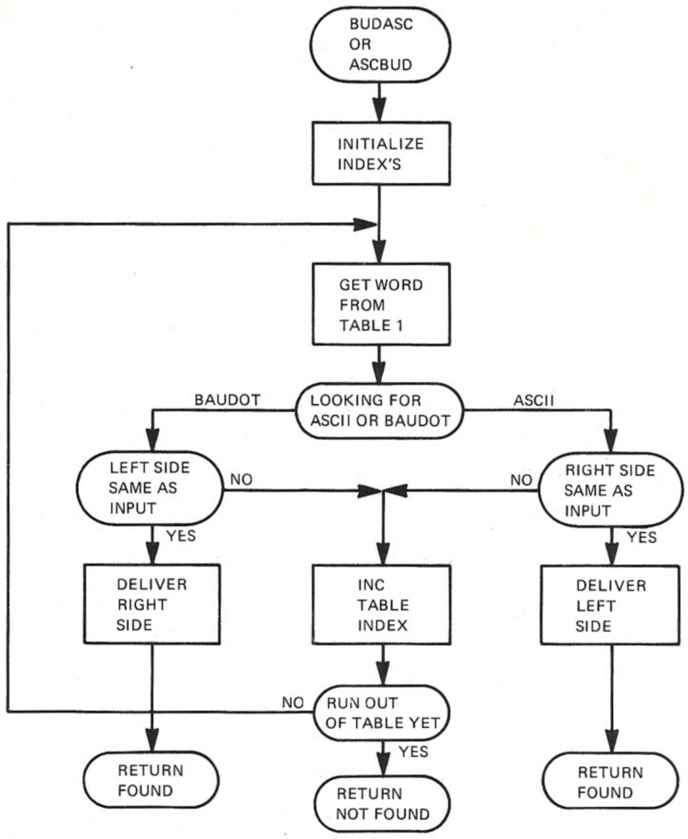


Fig. 3. Baudot-to-ASCII and ASCII-to-Baudot Converter.

the letters CONT (Control) on the top half and SPEC (Special) on the bottom. CONT and SPEC in combination with the LTR/FIG shift key allows me to form the ASCII control character. If the TTY is in LTR mode (FIG. SHIFT FLAG=0), the subroutine SPCVT is called to form ASCII characters that are not on the keyboard.

If the character was not a blank, the BAUDOT to ASCII converter BUDASC is called to form the standard convertible character and pass it on to the using software.

TIN

This routine (see Fig. 2) accepts a character from the keyboard by calling TIN (Type IN) and checks the incoming code for LTR or FIG shift. If it was either of these, it sets the shift flag for FIGs or clears it for LTRs, shifts the print mechanism, and then waits for the next character.

If the character was not a shift code, it delivers the character back to the caller (TTIN in this case) with the leftmost bit in the character equal to the shift flag. This is the only way the rest of the driver can distinguish between the letter "D" and a "$", which both have the same Baudot code.

BUDASC/ASCBUD Converters

This one piece of logic (see Fig. 3) converts both to and from ASCII by simple table lookup. The table is shown in Fig. 4. As you can see, the left byte of the table contains the Baudot character, the right byte the ASCII equivalent. Each Baudot character that requires a figure shift has its leftmost bit set, which causes it to be 200_8 greater than its letter shift cousin.

SPCVT

This routine, Fig. 5, prints a period immediately and then waits for an input by calling TIE. TIE is a simple routine that first calls TIN to get a character and then TOUT to print it.

Once input, the routine BADOCT is called in an attempt to convert the incoming Baudot character to its actual equivalent. BADOCT can be a table lookup or a series of subtractions from the octal values in Fig. 4. If it can't convert it, it assumes that this is the first character of a two character symbol (.GT. for ">", for example). It waits for the second character and then calls the two-to-ASCII converter TWOASC to generate the equivalent (Fig. 6).

TWOASC does another table lookup using the table shown in Fig. 7. If no conversion is possible, a "?" followed by another period is output and the process repeats. When a match is found, the trailing period is output and the ASCII character is delivered to the user. The logic on the right side of the flowchart allows you to enter three octal characters whenever you want to output any combination of eight bits. This allows you to make

up any of the nonprinting ASCII characters from your keyboard.

CONCVT

Control Converter. Control characters are normal ASCII characters with 100_8 subtracted from them. They are used for special functions in time-sharing systems and are generated on an ASCII TTY, such as an ASR33, by holding down the CNTR key and striking the second key.

This routine (Fig. 8) outputs a " / " on the printer and then waits for a character to be input. When one comes, it is echoed and the second " / " is output. 100_8 is subtracted from the ASCII equivalent of that character, and the control character formed is returned to the calling program.

Notice that it's not possible to hit a Baudot key that can't be converted to an ASCII code, so if BUDASC returns *not found* at this point, there is a bug in your program.

Output

The subroutine TTOUT (TTY OUTPUT) performs the reverse task of TTIN (see Fig. 9). It attempts an ASCII to Baudot conversion, and if successful, prints that character. If no equivalent exists, the routine ASCTWO is called to look for a possible two character equivalent. If one is not found, a "?" is printed, and the next character is examined.

TOUT

This routine (Fig. 10) takes care of printing the character, the shift requirement on the printer, and it also inserts dummy characters (zero) when a carriage return is called for. This allows the carriage on the printer to come to rest before attempting to print the next character.

Conclusion Problem 1

The use of two keys to form one ASCII character makes a Baudot machine ASCII compatible, and although not nearly as nice as a full ASCII keyboard and printer, the price is right! I picked up two 60 wpm machines for $15.00 from a ham friend whose wife wanted to park her car in the garage. The two 100 wpm MITEs I have were a little more, but together I have spent less than $100.00 for the bunch.

So, if you want hard copy and you're a ham, get on the air and ask around. If you're not a ham, look around in your neighborhood for a large antenna; they're nice people.

Speed Conversion

If you plan to use the driver program to talk only to software in your own computer (your assembler or BASIC compiler, for example), you have no problem. The computer will make the software wait while the Teletype prints at 60 wpm rather than the 100 wpm of an ASR33. It will be slower, but I never found it that bad. So, if that's the case, skip directly to the next section for the hardware interface (unless you want to know what a baud is or if you *think* you already know).

When thinking about the speed problem, it became apparent that I needed a buffer to store the data coming from my faster peripheral (I like to think of the time-share system as *my peripheral*) at 110 baud. It also follows that no matter what size buffer I built, under certain conditions, my 110 baud peripheral could overrun it.

One of the constraints then was "never ask for an operation, such as a multipage listing, that would overfill the buffer." A second problem was just how much slower than 110 baud is my 60 wpm printer; what about my 75 baud MITE printers; and what is the relation between words per minute and baud?

What's in a Baud?

I have read several definitions of this term, but it wasn't until I worked on this problem that I found one that really worked for me.

Sometimes a baud is defined as *a unit of signaling*

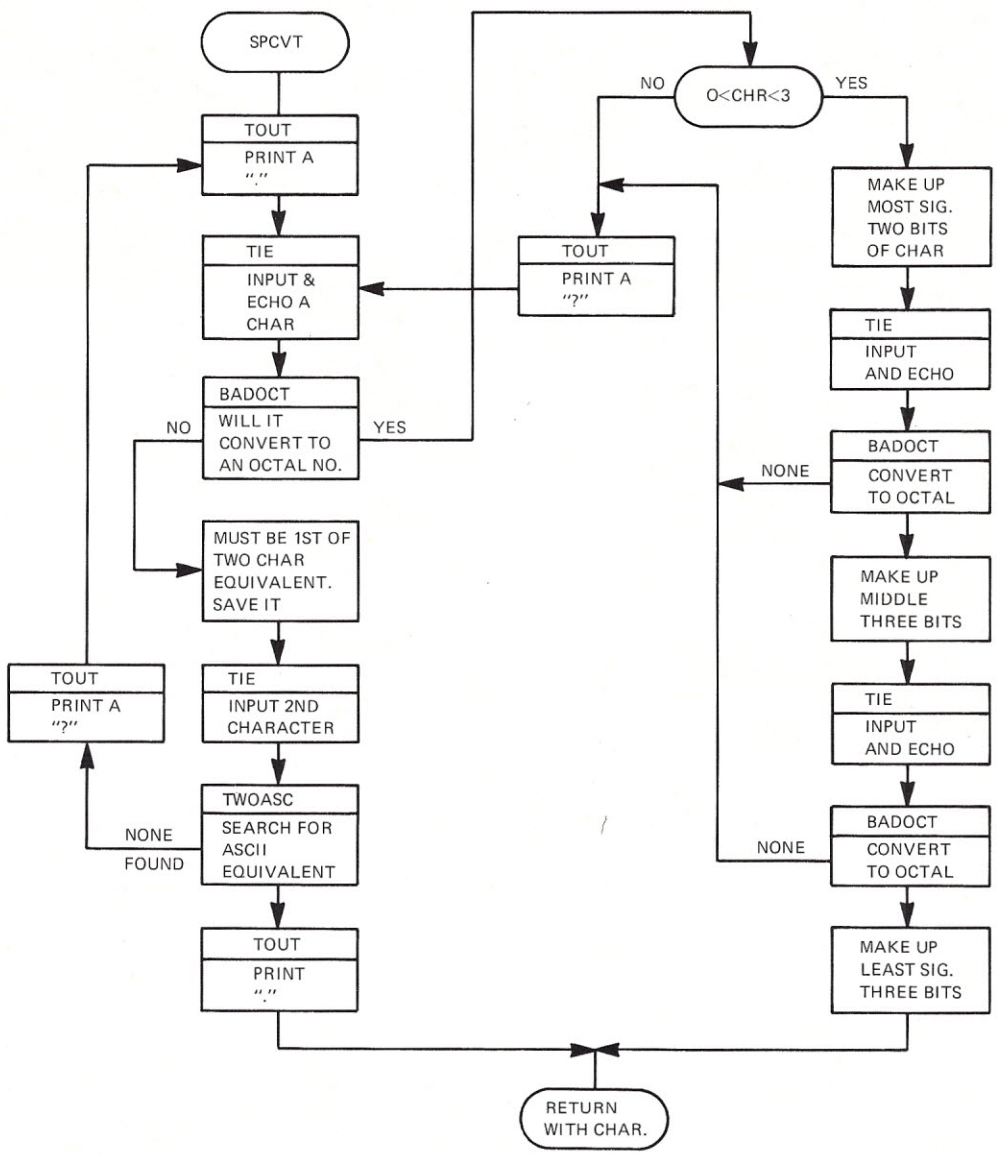


Fig. 5. Special Character Converter (SPCVT).

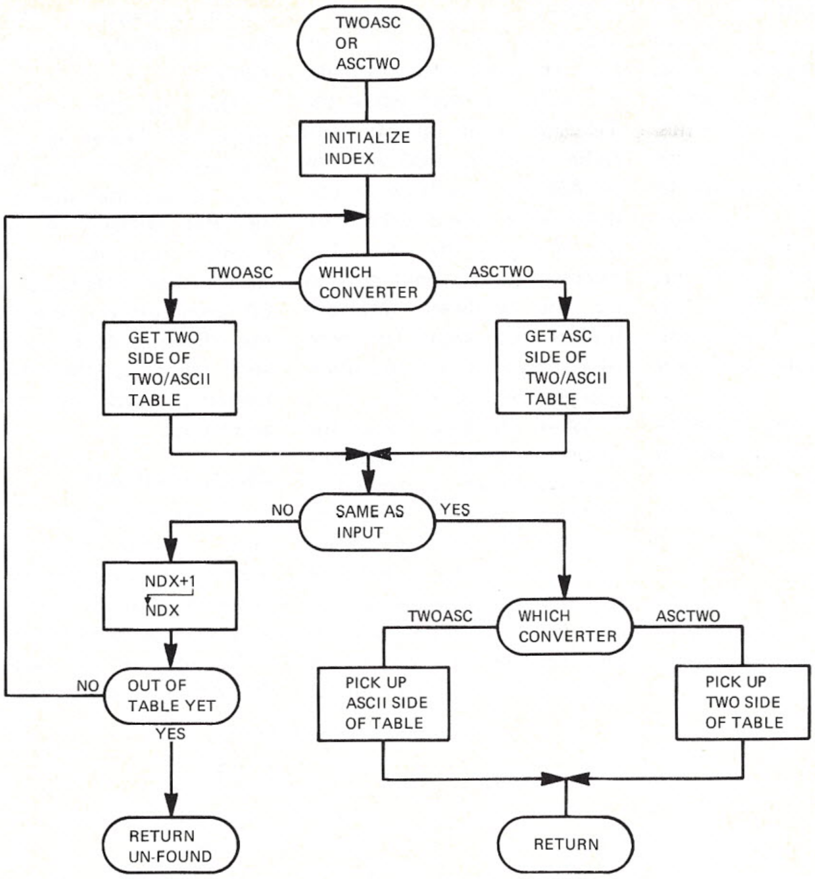


Fig. 6. Two/ASCII and ASCII/Two Converter.

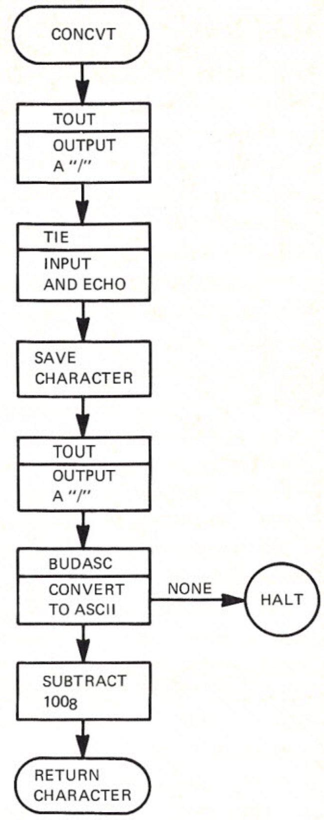


Fig. 8. Control Character Converter.

ASCII Character	and Code	TWIN BAUDOT Character	and Codes
(#)	043	(NO)	006, 003
(%)	045	(PC)	015, 016
(')	047	(AP)	030, 015
(*)	052	(AS)	030, 024
(+)	053	(PL)	015, 011
(@)	100	(AT)	030, 001
([)	133	(LB)	011, 023
(/)	134	(LS)	011, 024
(])	135	(RB)	012, 023
(↑)	136	(UA)	034, 030
(←)	137	(LA)	011, 030
(<)	074	(LT)	011, 001
(=)	075	(EQ)	020, 035
(>)	076	(GT)	013, 001

Fig. 7. Two Character/ASCII Table.

speed. Not much help there! At other times, it's the *number of bits per second.* This one implies that if I send one and only one bit down a line that, say, turns on a light, that the baud rate of the line is 1. This is regardless of the fact that the bit is one second wide or only a nanosecond wide. That definition does not help much either, does it! I read one that was something like, "a baud is the reciprocal of the period of the data bits in the transmission stream." This one is close, but it still leaves dangling the start and stop bits in a Teletype system. Are they data bits? If so, what about the fact that the stop bits can be twice as long as the data bits and in some codes even one and one half times as long?

The best way to think of the baud rate of a system is as *the reciprocal of the fastest bit on the system.* This makes baud rate equivalent to bandwidth, I know; but it's the one that works.

Fig. 11 was constructed for the common baud rates and word rates that I know. The starting point in construction of the table was the unit time (the fastest bit in the system) and the units/character. Notice that the DATA column contains 8 bits for most of the higher speeds. The eighth bit is a parity bit and generally set to zero when you are talking to a time-sharing service.

The logic for not transmitting parity is as follows: "If you send a character that somehow gets scrambled in the telephone lines or elsewhere, what could the time-share service do about it anyway?" It will echo the character back to your printer so you can then see that the printed character wasn't the one you keyed and you will know that an error has occurred and must be reentered. That is the one advantage to using a full duplex terminal.

The lower speeds are the ones that the Baudot machines use and no attempt at parity is made on them.

A *word* consists of six characters. This has no relationship to the real world words that you will type unless you happen to only type five letter words followed by one space. I think this rating of machine speed comes from the way telegraphy operators were speed rated. They were given five letter groups of *words* to send and receive and were rated on these *words per minute.*

As you can see from the chart, a 75 baud circuit appears to go a little faster than a 110 baud circuit. This is because of the smaller number of units in each character. In actual practice, the 75 baud circuit is slower because the operator must send shift characters each time he wants numbers or punctuation.

Ring Buffer

Now that I know my 45 baud Baudot machines are

about half as fast as the 110 baud circuit, and my 75 baud machine is just a little slower, what type of buffer should I use?

A simple linear buffer could be used, but I would have to write software that would allow the circuit to completely fill the buffer, then start my printer running until it was completely empty. This would increase the time on the line by better than two and, therefore, be very wasteful of computer time, telephone time, and my time.

I could have put in a double buffer where the circuit fills one while my printer empties the second, but I thought it would be more fun to implement a ring buffer. A ring buffer allows both the printer and the circuit to use the same buffer and have the printer *chase* the circuit around the loop.

Right away several problems appeared, such as: How does the printer know when it's caught up, and when does the circuit know when it's about to overrun the printer? Another interesting problem was the starting point when the buffer is empty. How do you tell the printer software the difference between *caught up* and *you're still at the starting line?*

A simple algorithm implemented in Fig. 12 and Fig. 13 shows how the setting and testing of pointers accomplish these jobs. The logic shown is quite simple; but since it did require some time for me to get it straight, I thought you would be interested. The routines require initialization when the system is restarted from scratch. The initial condition would be MT = 1 (Empty Buffer), FULL = 0 (it's not full), RED and WRT both equal to zero. The subroutines WRITE and READ were required for my machine because it's a 16 bitter. They examine the WRT or RED indexes to see if they are odd or even, then work on the right or left side of the word as required. If you have an 8-bit computer, the two subroutine calls can be replaced by a simple load or store.

In my system, the using software calls these routines. Control is then returned to the user at either call + 1 for operation complete or at call + 2 for ring full or ring empty.

Conclusion Problem 2

Ring buffers are fun to build. As an added feature, how about adding a heuristic trap that makes the ring bigger whenever the circuit catches up with the printer and starts losing data?

Computer/TTY Interface

The Teletype controller that I finally settled on uses a UART, two optical isolators, a 555 timer, and a handful of TTL chips. (See Fig. 14 for details.) The UART was chosen to take care of all the problems like start/stop bit generation, buffering, etc. This clever little chip combines in one 40-pin package all the logic that would take me 50 or so small scale TTL chips to implement. I use the Western Digital TR1602N, but there are many other pin-for-pin chips that will work.

The optical isolators were used to separate the signal line loop current supply and my computer power supplies. Although I have never tried it any other way, I have been told by those who should know that you may have *ground loop* problems if they're tied together. Not wanting any more problems than I already had and considering how cheap the isolators are, I use them in all my peripheral equipment.

The handful of TTL chips are mainly on the computer side of my interface; and since I am sure that it's different from yours, I won't go into those details.

All the logic shown on the left half of Fig. 14 is on a card within the computer main frame. The stuff on the right is mounted on a small

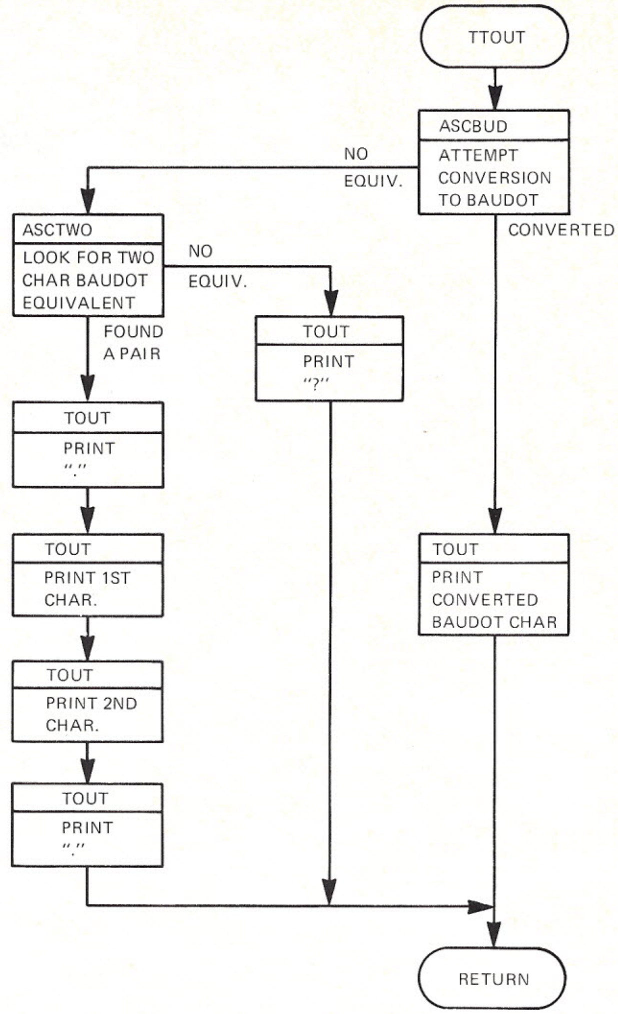


Fig. 9. TTY Output Routine (TTOUT).

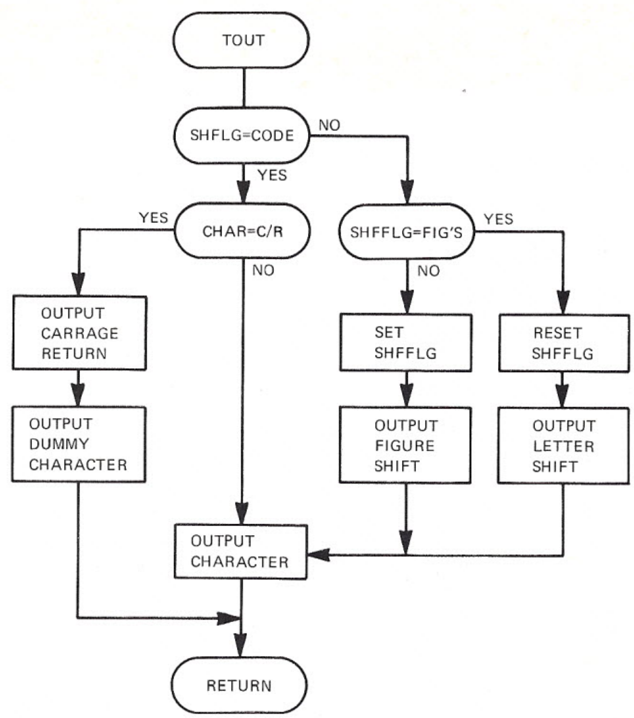


Fig. 10. Output a Character (TOUT).

BAUD	WPM	CPM	CPS	UNIT TIME	UNITS/CHARACTER START	DATA	STOP	UART CLOCK FREQ	PERIOD
45.45	61.2	367.2	6.12	22 ms	1	5	1.5	727.27 Hz	1375 micro sec
75	101	606.4	10.1	13.3 ms	1	5	1.5	1200 Hz	833 micro sec
110	100	600	10	9.09 ms	1	8	2	1.760 KHz	568 micro sec
150	136.5	819	13.6	6.66 ms	1	8	2	2.402 KHz	416 micro sec
300	273	1638	27.3	3.33 ms	1	8	2	4.804 KHz	208 micro sec
600	547.6	3285.8	54.7	1.66 ms	1	8	2	9.638 KHz	103 micro sec

Where:

$$\text{BAUD} = \frac{1}{\text{Period of smallest Bit}} \text{ or } \frac{1}{\text{unit time}}$$

$$\text{WPM} = \frac{\text{CPM}}{6}$$

$$\text{CPM} = \text{CPS} \times 60$$

UNIT TIME = GIVEN for each machine

UNITS/CHARACTER = DEPENDS on machine

$$\text{UART CLOCK FREQ} = 16 \times \frac{1}{\text{UNIT TIME}}$$

$$\text{UART CLOCK PERIOD} = \frac{1}{\text{UART CLOCK FREQ}}$$

Fig. 11. Circuit speeds.

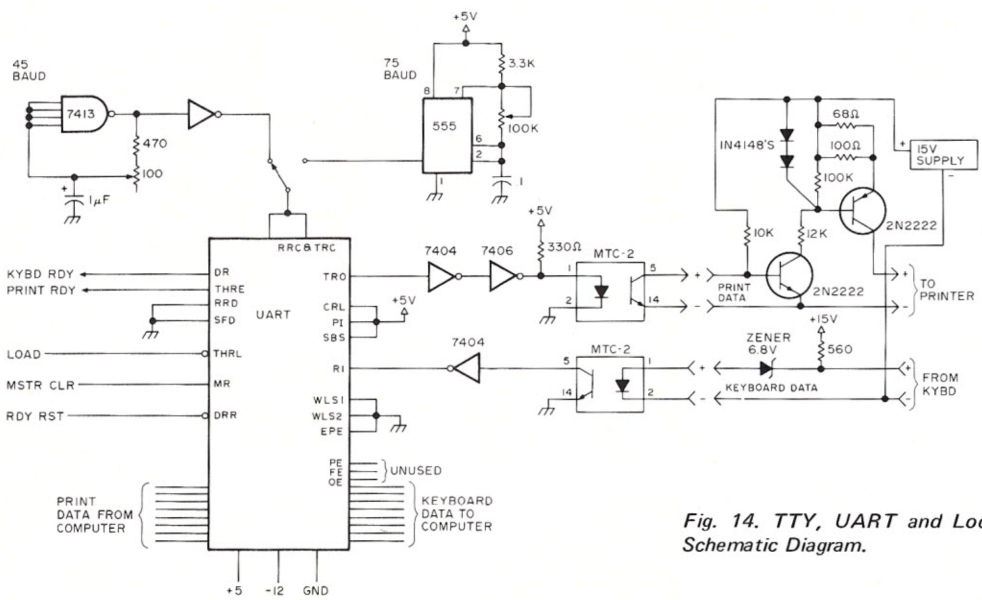
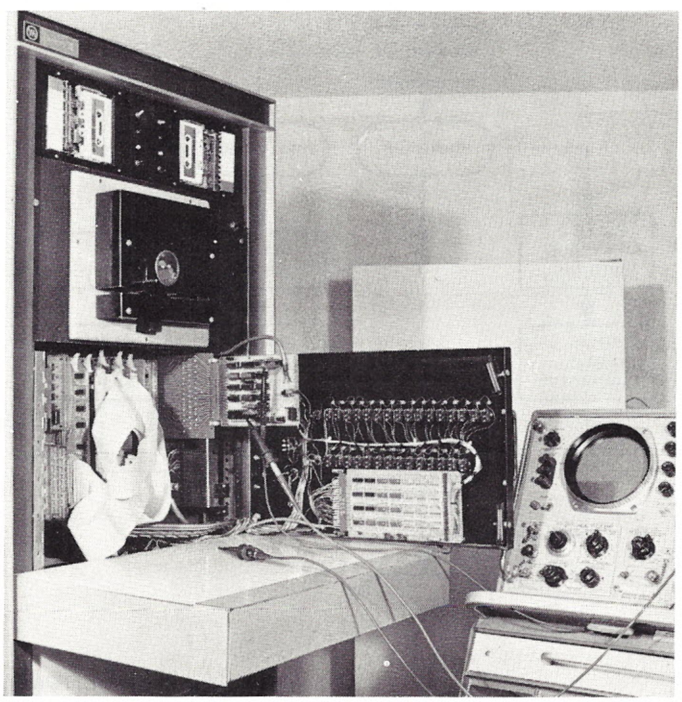


Fig. 14. TTY, UART and Loop Schematic Diagram.



vector card and placed within the TTY cabinet.

The loop current can be set to 20 mA or 60 mA to suit the particular printer by adjusting the resistor combination in the emitter circuit of the second 2N2222. The 68 Ohm/100 Ohm combination was just right for a 20 mA circuit.

I have a 25-foot cable between the Teletype and the computer, but I think it could go much longer if necessary.

I use only the left half of this circuit to drive my two MITE teletypewriters. Since they contain their own loop current supply and line sensor, the right half wasn't needed. The MITEs do require a small wiring change to allow the keyboard and printer to both use the same supply and still be able to run full duplex. If you have one of these machines and would like to use it in this manner, drop an SASE in the mail, and I will send you back a Xerox sheet showing the change.

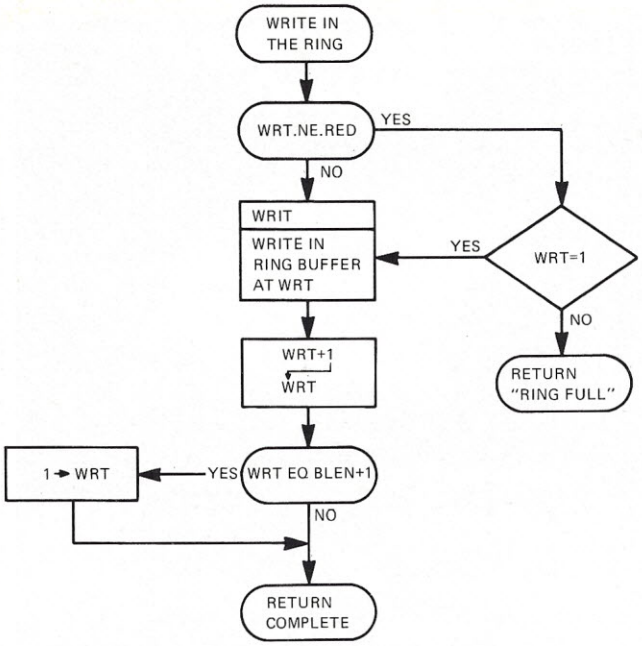
Since the MITEs can run at either 45 baud or 75 baud by a simple gear change, I put two UART clock generators on the controller with a switch between the two. I use a 7413 four input NAND for one of them and, for variety, a 555 on the other. With proper components, either circuit will generate a square wave that is stable and symmetrical enough for the Teletype.

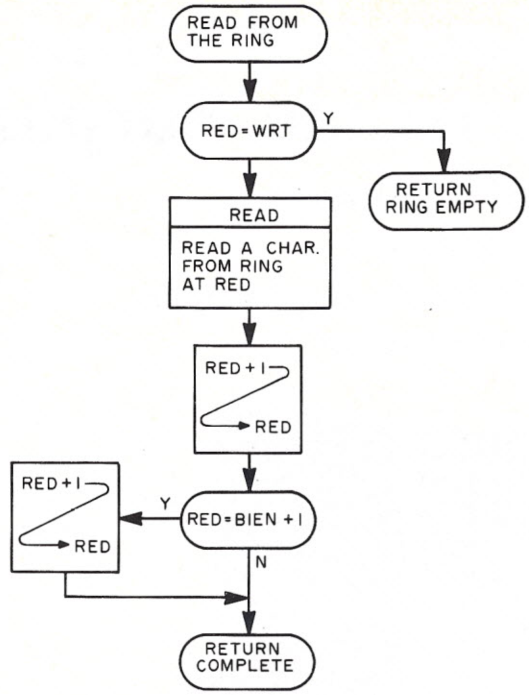
Conclusion Problem 3

The Teletype controller turned out to be a simple and very reliable circuit with none of the noise problems that I had been prepared to battle. The design of this circuit came directly from Ron Oehler of Control Data Corporation. Thanks Ron!

Article Conclusion

The addition of the TTY to my system has allowed me to take a light year jump over my previous non-I/O computer. I think that even if I had my CRT terminal running, with all its speed, I would still want to direct the output of dumps and assemblies to my teleprinters for





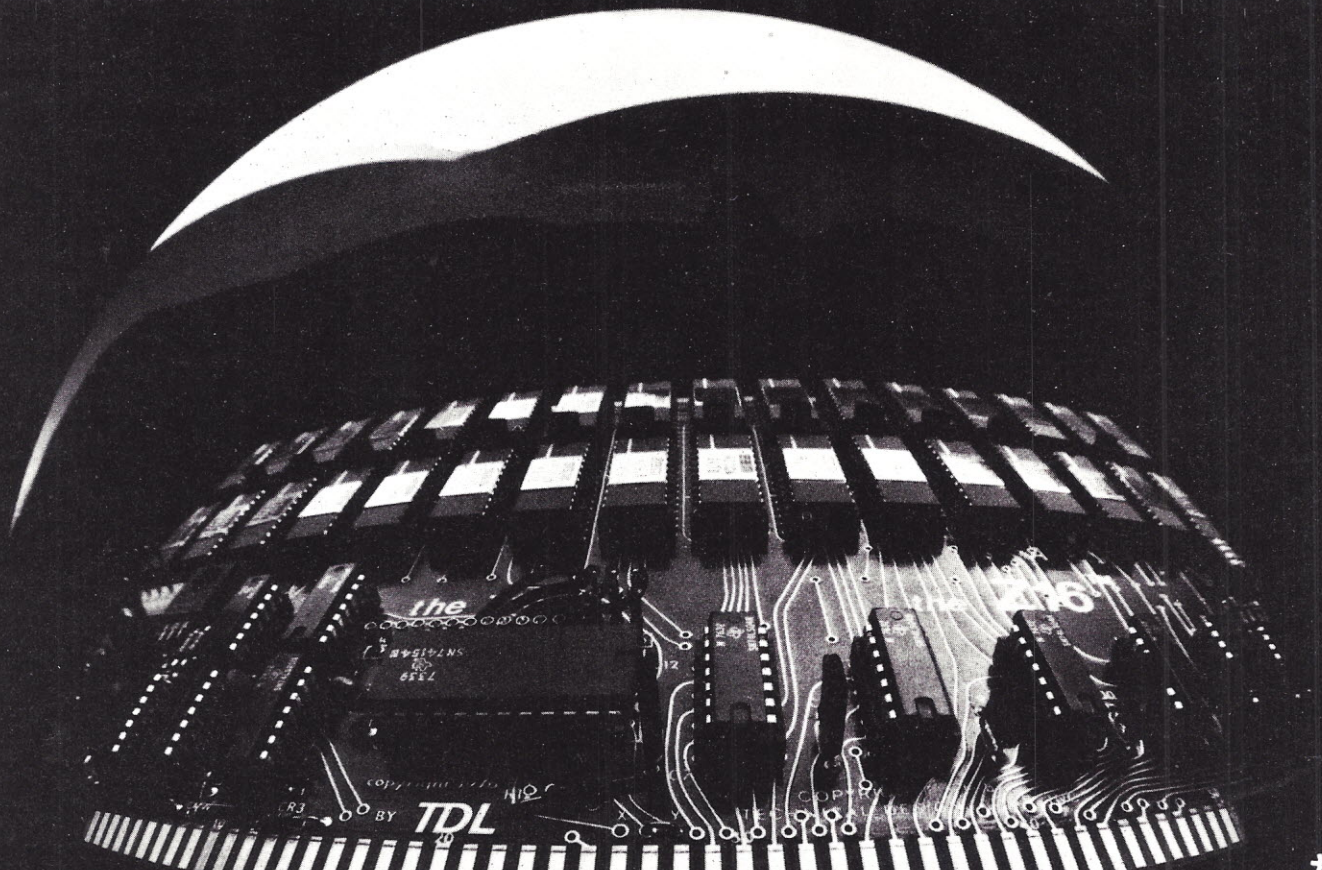
WHERE:
WRT=NEXT WRITE POSITION INDEX
RED=NEXT READ POSITION INDEX
BLEN=LENGTH OF RING BUFFER

examination of the data in a leisure manner.

Now that I have the conversion and computer to time-share problems solved, the next step is to connect my ham rig to the computer. This will allow a friend in a distant city access to the time-share system over his radio Teletype system (RTTY).

I also can build an automatic RTTY beacon that would carry on a conversation with a distant ham and keep a running log of all stations communicated with.

Good luck, and remember, it's only a hobby; when your relatives ask you, "When will it wash the dishes?", just smile and say: "Soon, very soon now." ■



MEMORY FROM THE FUTURE*

Z16

COMBINING:
- HIGHEST DENSITY
- FASTEST ACCESS
- LOWEST POWER USE
- HIGHEST QUALITY
- GREATEST VERSATILITY
- S100 BUS COMPATIBLE
- LOWEST COST 16K STATIC MEMORY MODULE AVAILABLE

Full 16K of memory on one card available in 4K increments. Buy only what you need now. Expansion later is easy with a board you have already tested.

Utilizes the EMM SEMI 4200 memory chip which is organized as 4K by 1 bits. Provides maximum access time of only 200ns. Added to board logic time, total board access time is below 250ns. No other memory board made to S100 bus specs can match this.

Each 4K block may be individually addressed at any 4K page border. You have the versatility of most 4K boards in your 16K package. Address changes are very easily accomplished by using a simple jumper scheme. Each 4K block may be individually protected by a switch.

Power consumption is outstandingly low! Only 205ma from the +8v, 105ma from the +16v, and 24ma from the −16v, for a FULL 16K. Battery backup with a simple jack connector.

Fully solder masked and silk screened board, sockets for all IC's. Complete documentation includes source code for comprehensive memory test program and paper tape of this program.

KIT: 4K - $169; 8K - $295; 12K - $435; 16K - $574; 4K expansion kits - $140.

16K assembled and tested: $644.

Delivery: Off the shelf to 30 days.

*OFF THE SHELF

Expand your memory. Hurry, use coupon below to order. Or call (609) 921-0321

Please send your Z16 MEMORY FROM THE FUTURE.

KIT: ☐ 4K - $169 ☐ 8K - $295 ☐ 12K - $435
☐ 16K - $574 ☐ 4K Expansion Kit - $140 ☐ Assembled & Tested - $644

Name ____ Street ____
City ____ State ____ Zip ____
I enclose ☐ Check ☐ Money Order - amount of $ ____
Charge Card Data: ☐ BankAmericard or ☐ Master Charge. # ____
Exp. Date ____ Signature ____
☐ Send COD. I enclose 25% deposit. ☐ Send your FREE CATALOG.

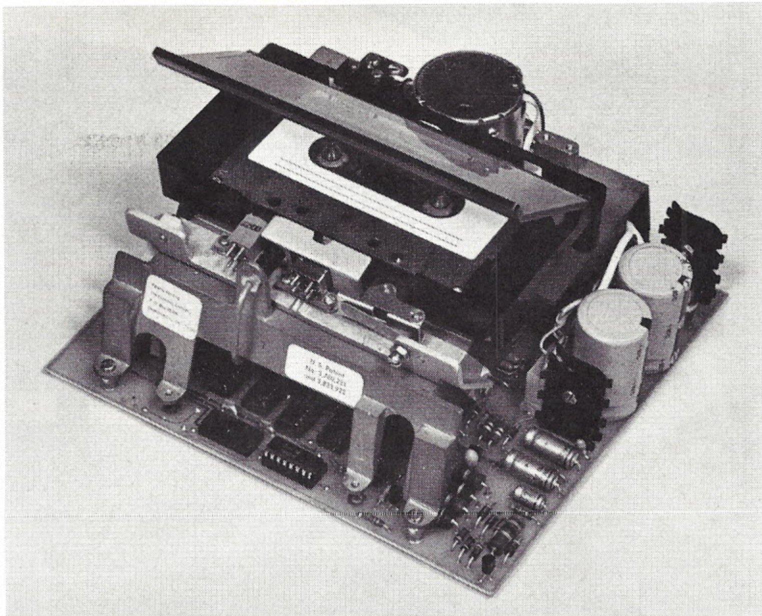
TECHNICAL DESIGN LABS INC. • Research Park • Bldg. H • 1101 State Road • Princeton, N.J. 08540

TECHNICAL DESIGN LABS
Z80

The Design Leader in μ Processing

TECHNICAL DESIGN LABS, INC.
Research Park • Building H
1101 State Rd. • Princeton, N.J. 08540

Art Childs
Sheila Clarke
Cybergrafix
P.O. Box 430
Glendale CA 91206



External Mass Storage

... Part 2: Digital and audio cassette systems

Part 1 of this article* discussed definitions and advantages of the use of external mass storage devices. To review briefly what that means to home computing, most microcomputer systems provide memory within the system that is volatile and limited in capacity. So we look to external mass storage methods for providing a means of remembering programs accurately in much larger capacities than internal memory can provide.

Virtues of mass storage devices and media, as opposed to internal memory, are ease of record handling, efficiency of repeated use of programs, accuracy, and of course a means to permanently store data.

Among those methods available are punched cards and punched paper tape, magnetic cards and magnetic tape, and magnetic disk. The three most used by hobbyists are punched paper tape, cassette tape and floppy disk. These three methods fulfill criteria established in Part 1 for appropriate devices to home computing. Those criteria are speed, ease of data recovery, volatility of media, capacity of storage, transportability, reliability, and economy. Part 1 discussed punched paper tape devices in depth and the following discussion will be devoted to cassette tape as a storage media.

Even as this article is being written, more and more devices are coming into the market. If we were to take the time to locate the various cassette systems and components and thoroughly evaluate each it's likely there'd be no time to write this article. Therefore, a few of the currently available devices have been chosen on the basis of availability of information and our familiarity with specific equipment.

Digital Cassette Recording Methods

In Part 1 of this article, we discussed the fact that while paper tape readers, and the media itself, are relatively inexpensive, punches are not. Other disadvantages of paper tape storage include the slow speeds of punches and readers, and that the tape cannot be reused. A faster method using a volatile media is cassette tape storage.

Two types of tape transports may be used for microcomputers: digital and audio. The single advantage of using an audio recorder over a digital is economy. Most hobbyists' systems have been designed to utilize the audio recorders, which are available at retail stores starting at prices from $30 up. Accessory cassette interfaces are also designed for use with audio recorders. Digital recorders, on the other hand, have two distinctive advantages over the audio type. Use of a digital recorder permits the magnetic tape to be completely saturated, or magnetized, giving the user 100% use of his media. More important however, is the ease of use through software control of external functions such as start/stop, etc. Audio recorders demand manual control, which is tedious at best. Therefore, using a digital cassette transport enables the user to closely emulate a floppy disk system, though it's slower and doesn't have the utility of a floppy.

Digitals are making their way to the forefront of hobby computing. The Phi-Deck is becoming well known and a unit is now available for

*See "The Trouble with Mass Storage Systems," *Kilobaud*, February, 1977.

The Phi-Deck is a variable speed digital tape transport which has a 4-motor control, remote control capabilities, fast start/stop, and fast rewind. Control boards are available for TTL, DTL, and CMOS compatible. Options also available for record/play and read/write electronics. Prices start at $100.

right around $100. For more information about the Phi-Deck range of tape transports, you can write to Triple I, Div. of The Economy Co., P.O. Box 25308, Oklahoma City OK 73125.

Audio recorders are the most popular now, but the variances in techniques have brought about standards such as the Kansas City Standard. This became necessary to make possible exchange of cassette recorded data from one type of system to another. Details of the KC standard may be found in *Byte Magazine*, February, 1976 issue.

The idea however is to record data serially, using the standard UART format at 300 baud, or 30 characters per second. Logic ones are represented by recording a 2400 Hz sine wave and logic zeros a 1200 Hz sine wave. When interfaced according to KC standard, the recorded data is then read from the tape and transposed into a selfclocking system which tolerates audio recorder speed variations of about ±30%. A selfclocking system in this case refers to the fact that the clock pulses are recorded on the media along with the data, rather than being generated by circuitry. Thus, when the rate of the data varies, as tape speed varies, so does the clock. So the effect of wide variations in speeds among most inexpensive audio recorders in most cases will be minimized. The standard also handles, to some extent, speed variations created by other electronics found from one computer system to another. How the data is to be organized on tape will vary between manufactured units, but can be controlled through the use of software.

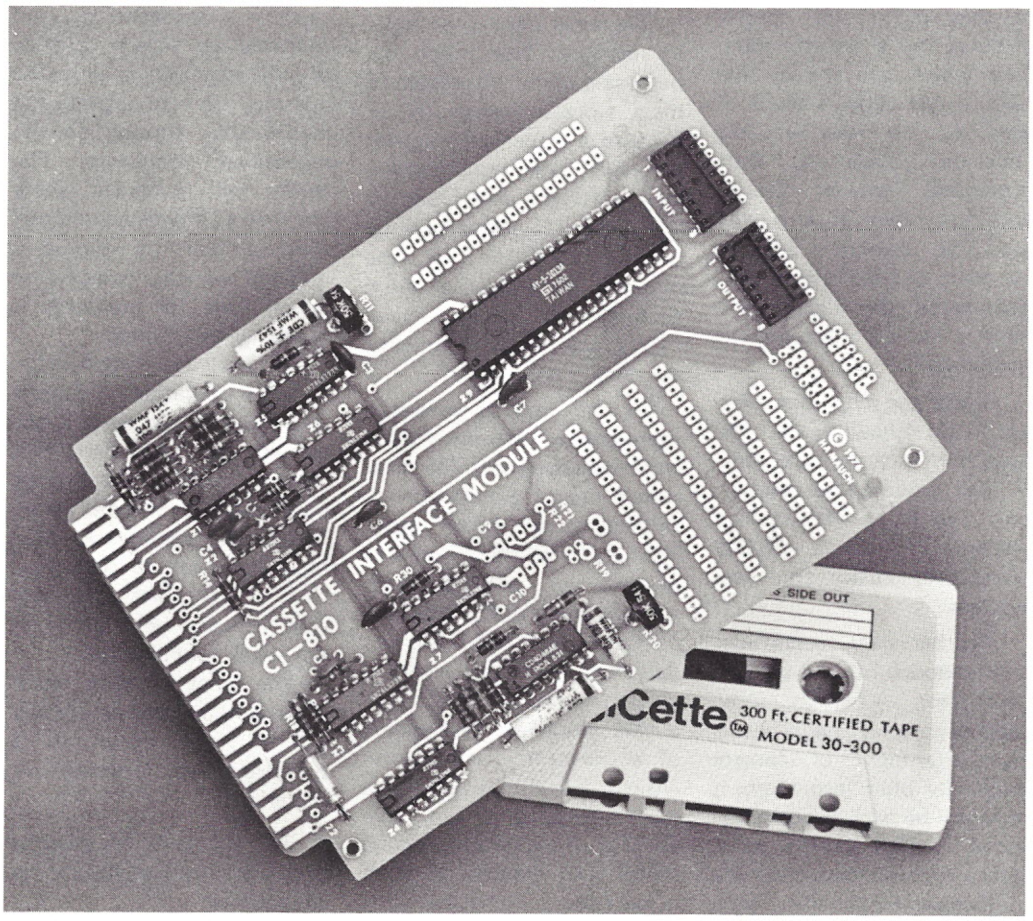
Various methods and types of circuitry are used by different manufacturers to convert an 8-bit byte of parallel data to serial data which may be recorded on an inexpensive cassette recorder. Just as methods vary from unit to unit, so does the speed. Generally, the higher the speed, the higher the price, and the higher the error rate.

Besides speed there are several other features to consider when purchasing a cassette system, not the least of which is starting and stopping the recorder. This feature isn't essential, but if you want to use your cassette storage for more than program retention, the ability to control tape motion via software is quite important. This feature will also add to the cost of the device. For instance, as discussed already, you can purchase a Panasonic audio tape recorder for about $40, but you'll have to manually control the start/stop, record, and other controls. Using a digital tape transport, such as the Phi-Deck, will permit remote control via software, and will cost $100 or more.

Another aspect of the recorder is the necessity of a parallel I/O port which must be considered part of the cost. Some units are made to plug directly into the Altair bus which eliminates the need for a separate I/O card. But at the same time it might mean the unit won't work with your computer. More and more of the personal computer manufacturers are offering a cassette board for their computer which solves the I/O port problem for you.

The recorder itself can be a critical component of a cassette system. Several manufacturers of cassette interface units recommend the use of very inexpensive recorders for low budget systems, particularly where speeds are kept low. For higher speeds, noise generated in the recorder becomes important, and for some systems (non-selfclocking) wow and flutter (rapid speed variations) in the recorder are critical. The advantage of a good interface, like the Tarbell Cassette Interface, is the ability to withstand a high degree of variation and allow the unit to perform reliably.

For the system that is to be used for more than loading and storing programs, a controllable recorder (facility for starting and stopping under program control) or transport mechanism, such as the Phi-Deck we've discussed, is essential. The reason is simply this. A manual recorder cannot be stopped under program control to allow data



PerCom Cassette Interface is Kansas City Standard, 8-bit parallel, priced at $59.95 assembled and tested.

Multi-Cassette Controller for 8080 CPU microprocessors is called the "Magic Black Box" by manufacturers, Ro-Che Systems. Will handle up to 4 cassette systems at a time for record handling. Write 7101 Mammoth Ave., Van Nuys CA 91405 for more information.

processing to be accomplished. Thus *all* the data must be stored in memory. A recorder with remote control features and a cassette interface which can utilize that feature will allow a block of data to be read in, the recorder stopped, data processed, recorder started, and next block of data read, etc. This capability adds to the cost of the system and, because of the added complexity, might decrease its reliability.

One of the major contributors to reliability is the quality of cassette tape used. A high grade audio cassette, one with a frequency response to above 10,000 Hertz will suffice for most systems.

Another item to consider is the possibility of using a controller. Although some cassette interface cards have a certain amount of control capability built into them, it may not be enough to handle more sophisticated tasks. Hence the possible need for a multicassette controller. We'll talk more about this in the section on cassette accessories.

The final item to consider is software. Not much has been said about this aspect of mass storage devices so far because a paper tape reader or paper tape punch driver (the software that controls the device) is generally quite simple. To utilize the inherent advantages of more sophisticated devices however, more complex programs are usually required.

In the case of cassettes, there are two extremes which we might consider. The most simple use of cassette is one termed *single block read.* That is, once the process of reading (or writing) commences, the process continues uninterrupted until all the data has been read into (or written from) memory. Such a case would be the loading of a program into memory. No subsequent action (program execution) can take place until the entire program is read into memory. So there is no need to stop and start the mechanical transport. For this type of operation, the data, or program, on the cassette is preceded by, and followed by several seconds of nulls (data ØØ) to allow the operator time to activate the loader after turning on the mechanical transport. The loader will recognize where the read is finished and signal the operator by typing a message on the console, at which time the transport is manually deactivated.

The software — or driver — for this operation is simple. Imbedded in the logic is the number of characters that must be read (or written), the address at which the program is stored, and checks that must be made to determine that the data was accurately read, usually referred to as the check sum algorithm.

The other extreme can be illustrated by considering that which must occur in a business upon receipt of material:

1. The *On Order* file must be updated to reflect the receipt of material.

2. The *Inventory* file must be updated to reflect the addition of the received material to the stock on hand.

3. The *Accounts Payable* file must be updated to reflect the need to pay for the material received.

Even if the computer being used had a full 65K of memory, it's unlikely that all three files, plus the program, could be contained in memory at one time; thus the need to read part of each file, stop the mechanical drive, process the data, start the drive to read and write more records, etc. The software to support such an activity could be written into each program, but that would amount to reinventing the wheel, and would therefore be inefficient. So, to enhance hardware, the manufacturers will often offer an *operating system* which is a piece of software to handle details of controlling the hardware and, in some cases, perform file management functions.

The devices chosen for presentation here represent a fair cross section of equipment currently available on the market. Many personal computing systems being offered now contain, as an integral part of the computer, a cassette interface, or the option for one. Most of the discussion of various aspects of cassette systems applies to these options. For those who are convinced that cassette is the mass storage device for them, one of these machines could represent a better buy if the option has the needed features. Some of these machines are:

Altair — Optional interface
Imsai 8080 — Optional interface
Poly-88 — Optional
OSI 400 — Optional
Apple-1 — Optional
Jupiter IIC — Integral dual
The Digital Group — Integral dual
Astral 2000 — Optional
Sol-20 — Integral

SWTP — Optional

TDL — Optional

Percom Cassette Interface

This device is one of the least expensive on the market. However, it does require an 8-bit parallel I/O port in the host computer. This fact will add to the overall cost of the cassette system, but enables it to be used with most of the personal computers currently in production.

The PerCom unit is designed to operate at 300 baud (KC Standard) and this would have to be considered a drawback, except for the fact that the speed may be adjusted upward by varying the size of a couple of components.

At the 300 baud rate, 8K BASIC will require about 3½ minutes to load. By modifying the circuit (according to manufacturer's specification) to run at 2400 baud, the same software would load in about one minute.

The PerCom interface is assembled and tested and sells for $59.95.

Tarbell Cassette Interface

One of the most popular cassette interface devices has become something of a standard, at least in Southern California. The Tarbell Cassette Interface is priced at $120 in kit form and is available at most computer stores. This device plugs directly into the Altair bus and can be used with either Tarbell Standard (187 bps) tapes or Kansas City Standard tapes (30 bps).

For those who wish, it may also be used at 540 bps. The high density formats, Tarbell 187 bps at 800 bits per inch, and 540 bps at 2200 bits per inch will allow much faster load times, but will require more care in the selection of tape quality. Still, to have the option is to cut 8K BASIC load time from 3½ minutes (KC Standard) to 43 seconds (Tarbell Standard), to 15 seconds at 540 bps.

Other features include four control lines to facilitate the control (start, stop, rewind, etc.) of the recorder, four extra status lines, and ability to adapt automatic digital cassette units and/or reel-to-reel tape transports.

Rating cassettes for hobbyist purposes had to be nearly pointless, as cassette has become the most popular form of mass storage. However, to be consistent:

Speed: Fair to good. Will load Extended BASIC in 5½ minutes to 25 seconds.

Ease of Data Recovery: Acceptable to good, depending upon the recorder and accessories (if any) used.

Volatility of recording media: With reasonable care, good. Bear two facts in mind: (1) the oxide will wear off the tape with repeated use and (2) if the tape is subjected to a strong magnetic field the data is gone!

Capacity: Good, particularly for the higher recording densities. A cassette storage case sold at record stores will hold enough games to keep you busy for many moons.

Transportability: Good. A cassette is small enough to slip in the pocket or the mailbox.

Reliability: Fair. A noisy recorder can cause unwanted glitches, but a well maintained transport and clean tapes can provide flawless operation.

Economy: Excellent to good, depending upon the accessories used.

Cassette Recorders

Numerous recorders are available that are more than adequate for the cassette mass storage system. We've already discussed a few of the features to consider, but here are some of the finer points you should be aware of. Characteristics to consider when choosing a recorder include frequency response (high end of 8000 Hertz minimum), speed variation and tape wear. For our home system, we have been using a J. C. Penny portable cassette recorder and a Tarbell Interface with good results. The recorder at $40 and the interface at $160 (assembled) makes it truly economical mass storage.

The digital recorder already introduced, the Phi-Deck, is rapidly becoming a popular device for enthusiasts who intend to make tape the prime mass storage mechanism. Due to its increasing popularity, interface manufacturers like Tarbell are including electronics modifications for the Phi-Deck in the interface assembly instructions.

Cassette Accessories

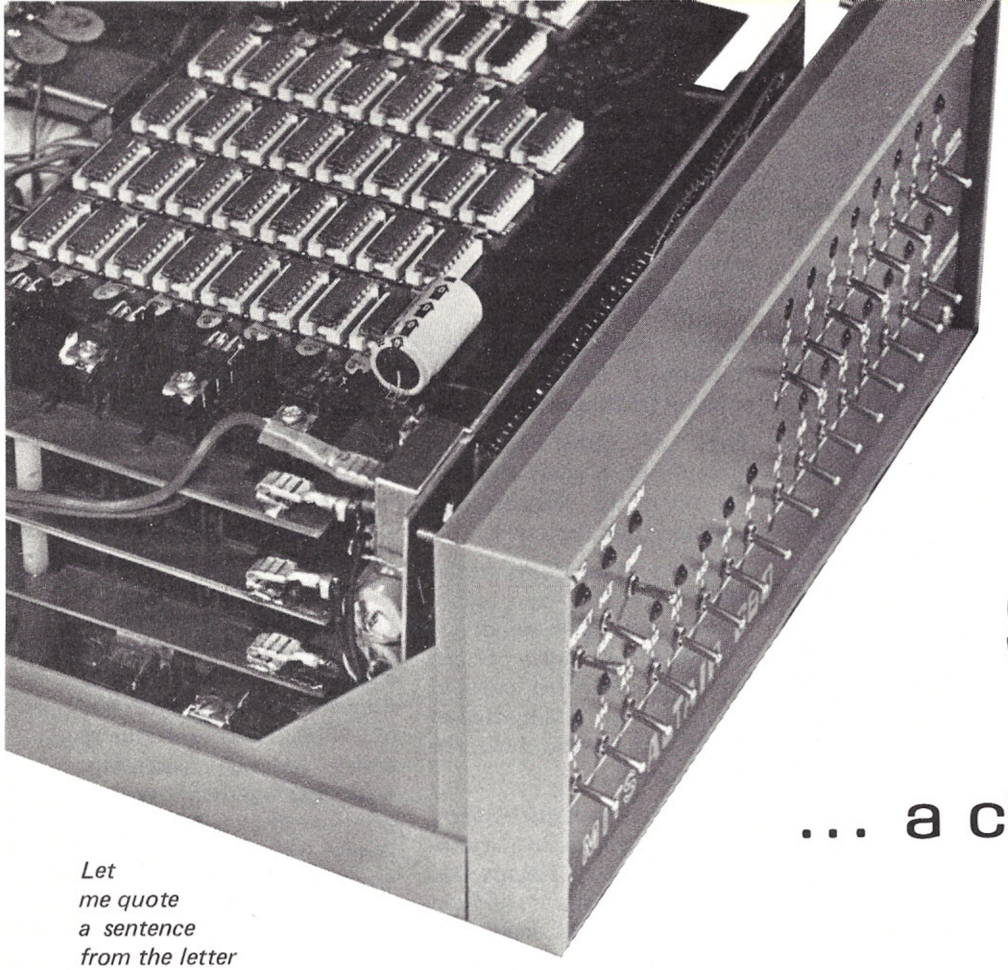
The Multiple Cassette Recorder enables the tape cassette user to increase the number of cassette systems for multiple file access. The Multi-Cassette Controller from Ro-Che Systems controls up to four cassette recorders with a Tarbell or MITS interface and handles records to be read or written. The unit, at $125 for four ports, comes completely assembled and with software. It may also be purchased in kit form, and in a 2-port configuration, either assembled or kit. ■

Table 1. Paper tape and cassette.

CRITERIA	PAPER TAPE	AUDIO CASSETTE
SPEED:	Read: — 10 to 300 bytes/sec. Punch — 10 to 120 bytes/sec.	Read & Write: 30 to 540 bytes per second
Time to load Ext. BASIC for execution	1 to 15 minutes (obj. in binary)	25 sec. to 5½ min. (object in binary format)
VOLATILITY OF MEDIA:	Completely nonvolatile, but cannot be reused.	Good: Can be destroyed by strong magnetic field.
EASE OF DATA RECOVERY:	GOOD	FAIR
CAPACITY:	FAIR: Extended BASIC in binary format requires approx. 80 feet.	GOOD: Several copies of ext. BASIC can be put on one 60-min. cassette.
TRANSPORTABILITY (media only)	GOOD	VERY GOOD
RELIABILITY	GOOD	FAIR TO GOOD
ECONOMY: Media Bytes/Dollar	4500	KC Standard: 1200 Tarbell Standard: 8,400 540 bytes/sec: 24,000*
ECONOMY: HARDWARE byte/sec cost / byte/sec	** Read: TTY $200 incremental readers incl. $150 allowance for interface approx. $150. Punch: TTY $200**	KC Standard $5.66 Tarbell Standard $.90 540 BPS — .32 Assumes $120 for cassette interface. $50 for audio cassette recorder.

* Assume a 60-minute cassette at $2.50.
** Includes both read and punch capabilities.

Note: Economy is expressed as bytes storage capacity per dollar for the media and cost per byte per second for the hardware, or cost/bytes per sec.



Make Your 680b Smarter

... a cheap memory expander

Let me quote a sentence from the letter Stu and Phil sent along with this article. "This scheme works so well and costs so little, we thought we'd share it with others." That's beautiful. It's a theme I've been harping on for some time. The more people we can get into this "sharing" frame of mind, the better. All too often, someone comes up with a good idea such as Stu and Phil have here but they just don't take the time to sit down and write about it. If you know someone like that, get on their case, okay. And if you have, or plan to get, an Altair 680, read on . . . you'll find this very interesting. — John.

Stuart Mitchell
14761 Dodson Dr.
Woodbridge VA 22193

Phil Poole
1408 Idaho
Woodbridge VA 22191

So you bought a MITS 680b and now can't afford additional memory? Well, this article will provide an economical answer with minimum work. MITS and their $685 offer of 16K memory sounds great . . . and maybe by the time you read this they will be in widespread use. In the meantime, for $325 you can have 13K "inside" your machine. The approach is simple — buy three 4K RAM boards from SD Sales, etch an extender board, mount sockets and jumper wire on the extender board, move existing 1K in accordance with MITS instructions, and plug in new boards. That is all there is to it. You are ready to operate.

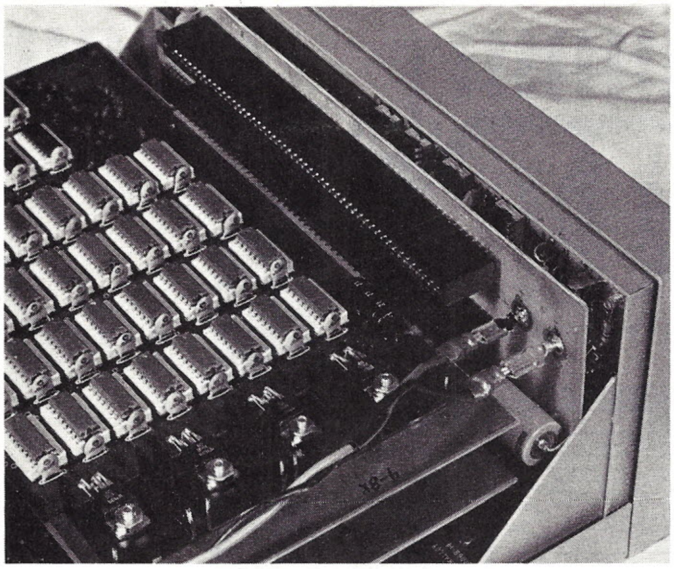
Theory

A review of the 680b shows that the data lines are bidirectional and buffered. The address lines are not buffered on the 680b edge connector. This leads to the requirement to provide buffering for address lines if more than 8K of RAM is to be added. When you examine the photograph of the installation, it should be obvious that you could add the RAM in 4K segments, that is, 4K this month, 4K next month, etc. SD Sales boards were chosen for the following reasons: on-board regulators and on-board buffering — inexpensive and available now! The 8800 Altair Data Bus is not bidirectional, but the SD Sales boards solve this with the 8T97 Tri-state™ noninverting buffers. The input bus and output bus are tied together and selection is provided by the inverted R/W line. It should be noted that the R/W line must be inverted to provide proper operation. The power supply in the 680b will run only one 4K RAM board so an additional power supply must be added for the second and subsequent RAM boards.

Construction

Well, you say, the theory sounds easy enough, now how do I do it? I suggest approaching the problem this way. Decide whether to etch or buy the extender card; then *do* it. Decide on a power supply option; either buy and rewind a small transformer and build it into the 680b, or mount an external supply if keeping it inside is a problem. Order the SD Sales memory boards. Three will fit inside the 680b nicely. Assemble the memory boards and put the jumpers and ICs on the extender card. Test carefully. Now for the details.

Etch the 1680 extender card using the foil layout shown in Fig. 1, or get someone else to do it for you. (See end of article.) Since cost was a large consideration, the 100 pin sockets used were fabricated by joining two wire-wrap sockets with 0.125 spacing. Normal soldertail sockets cost $5–$10 but the 56-86 pin wire-wrap sockets



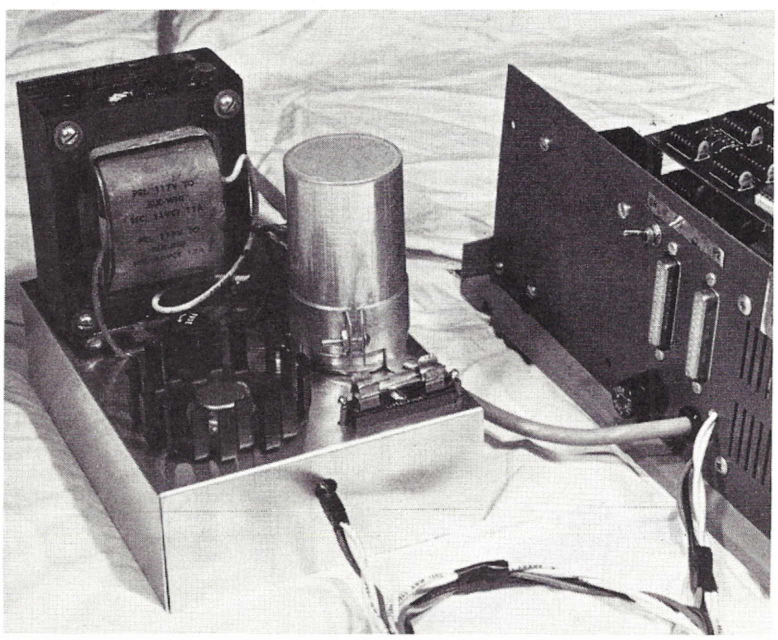
View of extender card showing socket detail and connections for the external power supply option.

surplus are \$.75–\$1.00. This makes it worth your time to join two together. Fabrication begins by sawing off the ends and filing or grinding down to the point where 100 pins (50 each side) have the proper spacing. Instant glue or epoxy the two sockets together and then solder them into the 680b motherboard and 1680 extender card. It may be easier to glue or epoxy after mounting the sockets on the extender card. Be careful to align the two sockets correctly. If you buy soldertail sockets, drill the holes to match spacing between rows of pins, as the spacing depends on the manufacturer.

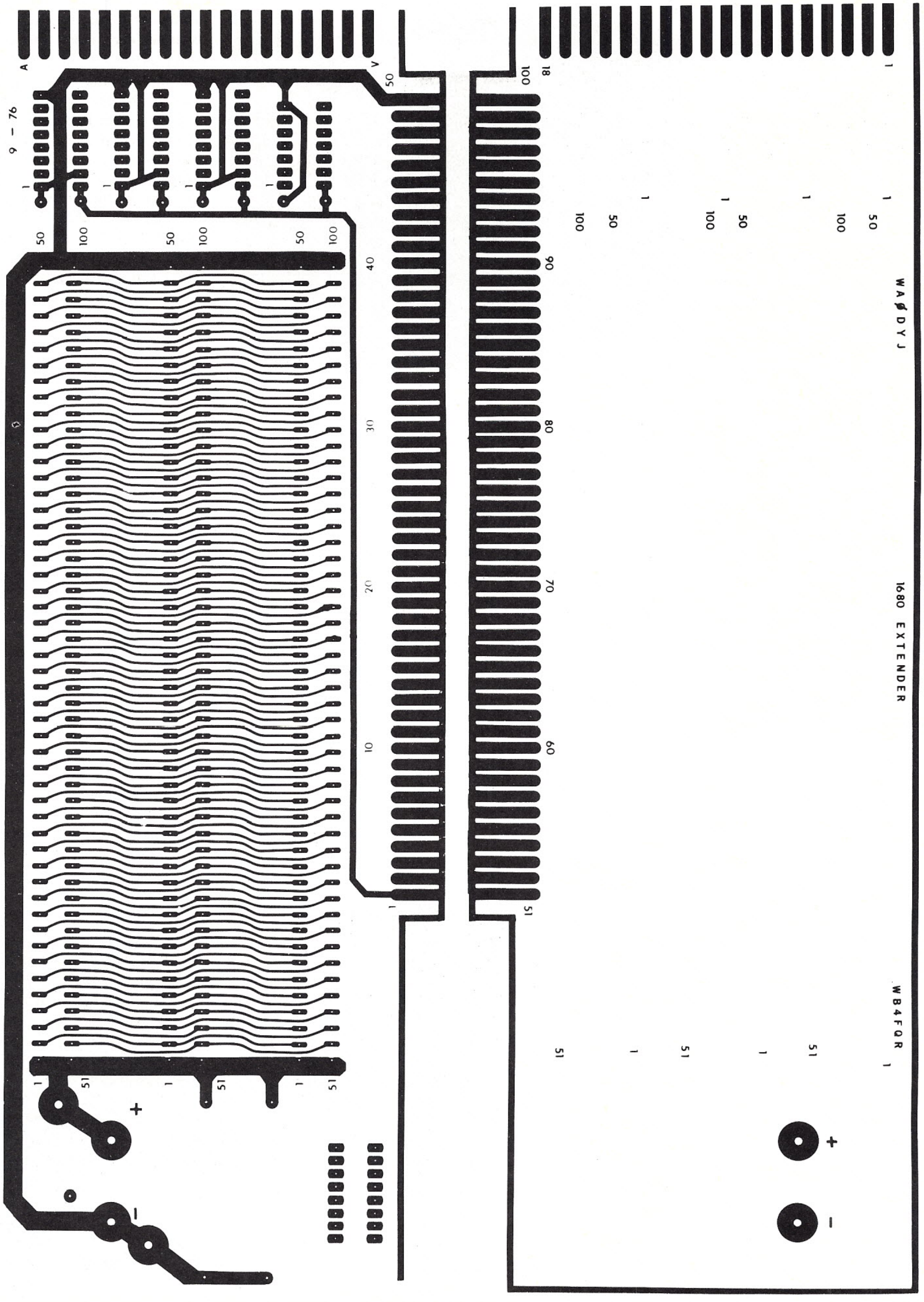
To keep the extender card simple, the 680b bus is cross-connected to the 8800 bus with small insulated wire and wired in accordance with Fig. 2. If several boards are to be used or you are extending to the outside of the 680b, buffering of all address lines is advisable. For 8K of memory, address buffering is not required if the SD Sales boards are used. Use sockets for the address buffers and the R/W inverter to facilitate testing. The photo shows the location of parts on the extender card. The 36-pin connector on the edge of the extender card is provided to connect to the outside, if desired, for additional RAM, video interface or for test purposes.

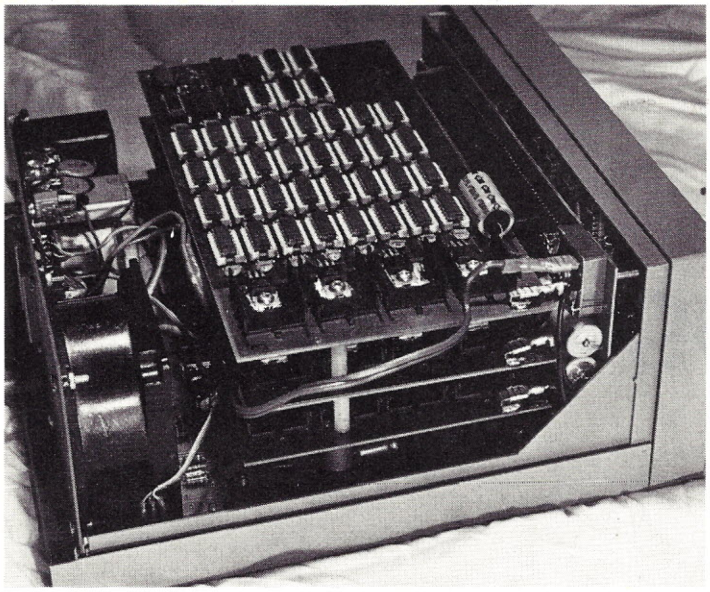
The power supply can be external or internal. Holes were provided on the rear panel by MITS for an extra transformer. A 10 volt, 4 Amp transformer is required. I made mine by rewinding a Radio Shack transformer as follows: Remove the channel holding the laminations. Pry loose the "E" and "I" laminations and remove all of them. Unwind the tape but be careful of the black wires as they are fragile. Unwind the secondary winding making sure to count the turns. If there are 160 turns and that is equal to 24 volts (6.7 turns per volt), then for 10 volts rewind with 67 turns of #18 enameled wire. After winding the new secondary, put tape over the windings to protect them. Be careful of the four leads when reassembling the E-I laminations. Be careful of the black tails and leave enough #18 wire to reach the extender card from the back panel. A very simple process that should take you only 20 minutes to do even if you have not done it before. The transformer mounts on the rear panel as shown. The bridge and filter caps are wired as in the photo. Note that each 4K RAM board is grounded to the main printed circuit board through pin 1,100 and also strapped next to the 3300 uF cap with a ground strap. This extra ground strap provides noise reduction on the 5 volt line. These lines may be soldered or you can use the automotive type slip-on connector. To complete the power supply, a full wave bridge and two 1000 uF filter caps may be mounted on the 1680 extender card as shown in the photograph. Since 2000 uF is not large enough, an additional 1000 uF is substituted for the 50 uF on each of the SD Sales boards. By using this approach, everything is mounted within the original enclosure.

If you choose an external power supply instead of building one inside the 680b, simply run heavy stranded wire from the external supply to the pads on the extender card as shown in the photographs. Eight to ten volts at 4



View of the external power supply. this one is 10 volts at 12 amperes.


1680 EXTENDER
WAØDYJ
WB4FQR
9 - 76



Side view showing the mounting of the filter caps and bridge rectifier on the 1680 extender card. Note the ground straps from the memory boards to the motherboard and the cooling fan in the rear corner.

Amps is required for all three boards. With either power supply installation, the fan must be in place and operating.

Testing

Alignment and testing is accomplished as follows: The

Fig. 1. PC board. (actual size, double-sided board). ← When etching, be sure you can read the printing on the copper foil.

680b memory should be relocated so it is not occupying the same space as the new 4K RAM board. Test the extender board without the ICs or the 4K RAM boards for continuity and shorts, an ohmmeter will do fine for this job. Make sure there is *no* continuity between the 680 pins 1 and 51 (5 V dc) and the memory boards pins 1 and 51 (8-10 V dc). This will ruin the 6800 chip plus your whole day if you screw up. Install the extender card with buffer ICs (if used) in place, and apply power. Ensure operation of the buffers; check that the power supply polarity to the memory boards sockets is correct on pins 1 and 51 and is, in fact, from the new power supply. *Note: 5 volt power from 680b is used only for inverter and buffer ICs.* Turn off power; insert one 4K RAM board; turn on power. Don't forget to jumper the addresses on the memory boards. Check that each bit can be stored in a location to ensure the data bus is OK, and then check each address line separately to ensure address operation. The 680b memory test can be located in the relocated 1K and used to test the new RAM boards. After the first 4K RAM checkout, add another 4K RAM board; test the data bus; next test the address bus; and then test the last board the same way.

Now you have either 5, 9, or 13K bytes inside the 680b box with minimum expense and just a little time. If you want additional memory, the 36-pin edge connector is available for this purpose. Add the jumper wires from the buffers and data lines and use a ribbon cable to connect to the external memory. Try to keep the ribbon cable length to a minimum. It may be advisable to run a heavy ground wire between the external unit and the 680b to keep noise and external interference to a minimum.

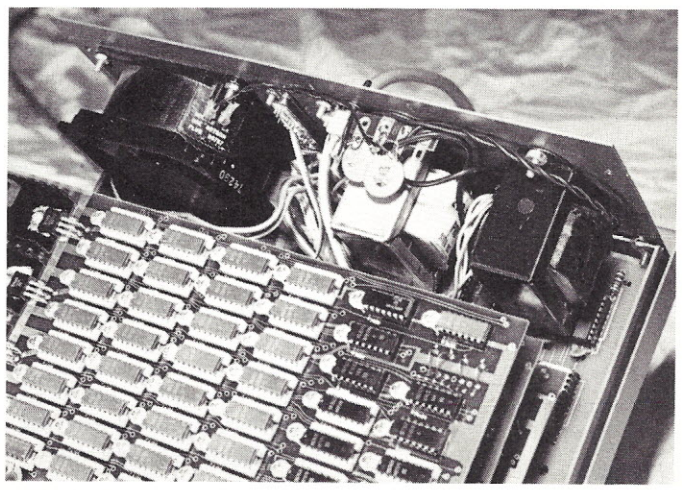
Results

To date, we know of three 680bs with this extender card scheme installed, up and running. One has 5K using a Baudot TTY, one 9K with a 33 TTY, and one has 13K using a video terminal. All used the SD Sales boards with no trouble encountered. External expansion is underway currently.

In conclusion, we would like to thank John Hoffman WB4GJZ for the photographic work and invite your comments and/or suggestions. Full size negatives, positives or PC boards are available from the authors. The PC boards are etched and plated but undrilled for $11, the artwork is $5 for negatives or positives. Some sockets are available at 8 for $5. Please specify. ■

680 Designation	680 Pin #	Buffer In IC#Pin#	Buffer Out IC#Pin#	SD Sales Pin#	SD Designation
GND	50/100	–	–	50/100	GND
+5	1/51	–	–		
BDØ	27	–	–	95/36	DOØ/DIØ
BD1	28	–	–	94/35	DO1/DI1
BD2	31	–	–	41/88	DO2/DI2
BD3	81	–	–	42/89	DO3/DI3
BD4	90	–	–	91/38	DO4/DI4
BD5	42	–	)	92/39	DO5/DI5
BD6	43	–	–	93/40	DO6/DI6
BD7	44	–	–	43/90	DO7/DI7
R/W	14	7404/3	4	68	M Write
R/W	14	–	–	47	S Memr
		if used			
AØ	71	8T97/12	11	79	AØ
A1	19	14	13	80	A1
A2	17	10	9	81	A2
A3	15	2	3	31	A3
A4	23	6	7	30	A4
A5	12	4	5	29	A5
A6	11	8T97/14	13	82	A6
A7	9	12	11	83	A7
A8	53	10	9	84	A8
A9	7	6	7	34	A9
A10	3	4	5	37	A10
A11	4	2	3	87	A11
A12	25	8T97/14	13	33	A12
A13	56	12	11	85	A13
A14	30	10	9	86	A14
A15	32	6	7	32	A15
ALTAIR 680 BUS				ALTAIR 8800 BUS	

Fig. 2. Extender Interconnects.



Detail showing the transformer mounting on the rear panel.

Stop Bugs Now!

... take time to design your next program

Programming is not an art (being generated by artists) but is much closer to being an exact science which requires specific steps and procedures to get it done right. Tim discusses those steps and procedures in the following article ...and also throws in some tidbits on structured programming which you'll find interesting.
—John.

Tim Barry
505 Cypress Point Drive #206
Mountain View CA 94043

How often have you heard or repeated the following question: "I understand the hardware OK, but how do I learn all this programming stuff?" This question and its countless variations are heard at hobby computer clubs and professional engineering conferences alike. The fact of the matter is that the microprocessor has thrust programming squarely upon those who thought they needed it least — the hardware designers. Like it or not, fundamental programming skills are going to be needed in most electronics design from now on. This article describes one basic available to the would-be programmer.

This article is based on my design of an existing product — *Microcomputer Programming with Modu-Learn**, a self-teaching software course for new microcomputer users published by Logical Services Incorporated, Mountain View, California. I have embodied in that course the techniques and approaches described in this article. As a professional programmer, I think this approach offers the best compromise for new users learning to program small computers.

* Registered trademark

Why Learn to Program at All?

At the Personal Computing Conference held in Atlantic City in August, 1976, I asked a prospective computer hobbyist if he intended to learn to program. "No," he replied, "I'm going to use the hardware." Alas, our hapless new computerist is in for a rude surprise. You want to use the hardware, you learn to program. Period. His response, however, is a starting point for discussing why people learn to program.

People end up in programming for two basic reasons: they wanted to learn or they had to learn. In the former category we find most professional programmers or those aspiring to that field. The actual design and implementation of the programs is where these users find satisfaction. In the later category we find most hardware designers learning microprocessors, hobbyists who bit for a computer and now need to do something with it, and in general those who need programs to make a computer do something they need. These two groups usually approach learning how to program in fundamentally different ways.

The person interested in learning to program for the sake of programming approaches learning with a student's eye. He plans a career or a serious hobby, and he knows this takes time. He usually studies for years to learn the craft, and he remains a student through the trade press and his day-to-day work. The lore and technology of software interests this user and he will study both to advance his knowledge.

Now our user bent on using programming as part of his system is not likely to be so patient. His interest is concentrated in other areas, and he only wants enough programming to get his job done. "Forget all that theory jazz, this thing has to run by next week."

The irony of this situation is that both types of programmer need to approach the design of their programs in the same way. Their motivations may be different, but

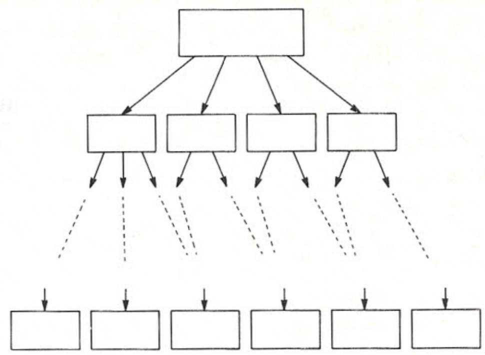


Fig. 1. Problem Partitioning.

the surest way to get good results is a systematic procedure which has been empirically developed by observing the best programmers at work. The professional programmer is very likely to be taught this way, particularly if he is entering the field now. Other programmers need to take the time to learn this approach because it offers the best chance of turning out working software in the shortest amount of time.

Strategy and Tactics

The approach I advocate clearly separates the logical design of the system from the way it is ultimately implemented. The situation is roughly analogous to military operations. The strategic planners set up the goals and priorities that are desired. The tacticians then set out to use available resources to see to it that these goals are met.

By deferring implementation-dependent details until well into the design cycle, we are able to concentrate on the big picture. This gives us the best chance of coming up with a total system which has been well thought out and really made to solve the problem. Once we have a good problem definition we can pick the language or processor best suited to solve it. The procedure breaks down into six basic steps:

1. Define the problem.
2. Break big problems into parts.
3. Develop the algorithms for each part.
4. Write the programs to implement the algorithms.
5. Test and interface the programs.
6. Final test and document the final system.

As you can see, we start out with the big problem, whack it up into lots of smaller problems, solve them, and rebuild the parts back into solution to the original big problem.

This procedure is certainly not new. We use variations of it to solve almost all engineering problems. However, the techniques required to make this approach work for programming are quite new.

For many years programming has been regarded as more art than science. Unfortunately, artists are not necessarily the best people to whom to assign projects which must meet schedules. The history of programming is filled with horror stories of programming projects which cost five, ten, even twenty times their projected budget. The systematic approach developed as an attempt to make programming a more controllable process. Unless personal computer users take heed and learn from previous experience, they are going to add more expensive (and frustrating) chapters to computer history.

Defining the Problem

It is not possible to solve a problem if you don't know what it is. Since this is obvious, why do so many programmers start their problem solution by sitting down and writing source code? Usually because they think programming is somehow different from other modes of problem solution. Wrong. Software is designed; before you even think about the program you have to think about the problem.

The amount of definition you will need for your problem will vary with a number of factors. A language processor for retail sales will need a larger specification than an I/O driver for your home system, but they both require definition. This definition usually evolves over the course of several attempts.

One thing is certain: you must write down your problem definition. It is not enough to simply think about it for a while. The process of putting the words on paper is essential to your arriving at a clear statement of the problem. Write down *all* operating characteristics, input formats, output formats, error conditions, and everything else. Strive for a complete description of how the program will look to the outside world, regardless of how it will look on the inside.

A good problem specification includes all the features that are to be in the final system. This means that once the spec is done you stick to it. It is usually the kiss of death when "neat," unspecified features start to creep into the system during implementation. If you want neat features, design them in from the start. Think about the problem and make sure the specification reflects the best possible solution; then use it.

It takes time to write a good specification, but it will save you time later. A good rule of thumb is that your specification is complete enough when you can hand it to another programmer who had nothing to do with the definition and have him implement the system. If the program he writes works the way you intended, the specification was OK. Few specifications are this complete, but you can at least try. With some attempts I've seen trying to pass as specifications you'd be lucky if you recognized any resemblance between the definition and the implementation.

Divide and Conquer

Once you have a good problem definition you can begin working on the solution. It has been well established that it is not effective to work on the whole system at one time. The human mind being what it is, the best solution is to divide the problem up into a series of progressively smaller modules. You repeat this process until you have isolated all program functions into chunks you can handle.

When first learning you will no doubt break all programs down into very small pieces. As you gain experience you will develop a feel for which modules will require the most work, and you can adjust your level and detail accordingly. However, the basic rule should always be: when in doubt, reduce it further.

It is easy to say "divide that system into pieces." It is

quite something else to actually decide where to cut. Most problems are stated as single entities and they often appear to be virtually indivisible. Not only that, the same system can usually be divided several different ways, and each way will produce different implementation problems.

Your guide in partitioning should be the specification. If it was properly done, there should be a large number of system functions described. Each of these unique functions will be a major system block. Some commonly encountered major blocks are input processing, function computation, data transformations, output processing, error handling, etc. Each of these major blocks is then further subdivided. Thus error handling might break down into input errors, overrange errors, computation errors, and so on. These blocks themselves may be further divided. An overall diagram of the partitioning process is shown in Fig. 1. At each level the scope of the block becomes smaller and the system details begin to take shape.

Algorithm Development

Once you have the system broken down into blocks, you begin to develop the logic required to make each piece do its thing. Note that we still haven't mentioned the computer. You should be able to do all of the blocking and algorithm development independently of the language and computer ultimately used for implementation.

The key to algorithm development is control of the sequence in which operations are performed. A computer can only do one thing at a time. The development of algorithms is, therefore, a study of "do this," "now do this," "if this then do that," type statements. This is exactly how you develop your algorithms.

There are two basic approaches to representing algorithms. One way is to use the familiar flowchart. Flowcharts use a small set of symbols to represent the various operations performed by an algorithm. They have been around a long time and they're undoubtedly here to stay.

An increasingly popular alternative to flowcharts is the use of a meta-language. This is a means of representing an algorithm in a language which is a cross between English and an actual programming language. Proponents of meta-language feel that developing an algorithm in words makes it easier to understand than the terser flowchart symbology. I think it's largely a matter of personal preference. If I know in advance the program is to be implemented in assembly language, I tend to use flowcharts. Algorithms destined for higher level languages implementation usually end up in meta-language. What is important is that you use something to symbolically define your algorithms.

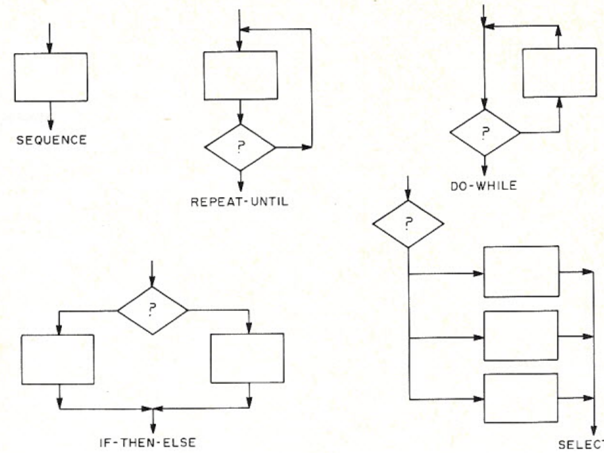


Fig. 3. Structured Program Structures.

Fig. 2 shows a flowchart and meta-language representations for a simple routine to form the sum of the first 100 integers. The flowchart gives a more graphic representation of the process, while the meta-language (very similar to PL/I) allows you to represent the algorithm almost as you would describe it to another person.

Whatever method you use to represent your algorithms, the development process is basically the same. First, define the inputs and outputs from the section. Next, establish the sequence of events that uses the input data to derive the output data. Finally, refine the algorithm to a level of detail which is adequate for program implementation.

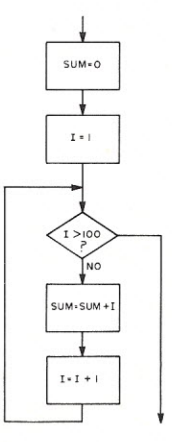
After you have developed an algorithm you should test it to be sure it works as you intended. You need to be sure it can correctly handle good data, bad data, and marginal data. Most program logic errors arise from inadequate algorithm testing during development. It is much easier to test algorithms in general logic form than it is to debug programs. When working on the algorithm you can concentrate on its operation without worrying about the many details imposed by actual computers and languages. It is a waste of time to commit to a program any algorithm you are not convinced is completely correct. Part of the algorithms you think are perfect will have bugs anyway, so why compound the problem by starting with programs you know have problems.

Algorithm Implementation

Once you have defined algorithms for the system, you can begin the implementation phase. Here is where we shift from strategic planning to the tactics required to make the algorithms run. This requires you to have a good understanding of your target language or computer. Each individual system has its own idiosyncrasies, and these are where most implementation problems arise. To minimize these problems you should try to limit your implementations to a fairly restricted set of program structures.

Structured programming is called many things, not all of them favorable. Without going into a long tirade, let me state the basic rationale that underlies the use of structured implementation. Hopefully this will be sufficient to convince you to consider learning how to use structured programming.

As we stated earlier, programs are a collection of operational sequences. As control passes through the program, these operations are executed one after another.




```
/* CLEAR SUM, INITIALIZE COUNTER */
SUM = 0
I = 0
/* LOOP UNTIL DONE */
LOOP: IF I > 100 THEN GO TO EXIT
        SUM = SUM + I
        I = I + 1
        GO TO LOOP
/* EXIT FROM LOOP */
EXIT:
```

Fig. 2. Flowchart and Meta-language Comparison.

Over the years it has been determined that many program errors result from the arbitrary transfer of program control from one section to another. These transfers of control usually result from unrestricted use of program branches. Structured programming eliminates this problem by curtailing the use of the program branch.

In structured programming we build programs from a limited number of simple control structures. These structures have well defined entry and exit characteristics, and the operation of program sections which use them can be directly verified for correctness. These structures limit themselves to a single entry point and a single exit point. This means that program sections constructed using them tend to flow into each other rather than jumping all over the place.

Another advantage of a structured implementation is that repeated use of the structures makes you very familiar with them. You will build up a repertoire of standard operations and this in turn saves you from reinventing the wheel every time you need a common function. Use of previously debugged and tested program sections can both save you time and increase program reliability.

Not all is perfect with structured programs, however. There is no such thing as a free lunch. Structured programs tend to be a bit larger and less efficient than some highly fragmented and optimized programs. Also, the limitation to controlled structures does not sit well with some who feel unrestricted programming is part of the art form. However, increased development speed, lower debug time, and greater modification ease usually far outweigh any small penalities in efficiency and memory size.

Some typical structures are shown. in Fig. 3. Other structures are also in use, although these are probably the most common. Note how each structure has a single entry and a single exit. In practice you will construct your algorithms using the generalized logic forms of these structures. These general logic forms will then be implemented using the machine or language specific structures you have available.

A really rigorous application of structured programming almost always requires you to work in a higher level language. Even then the constructs you require may not be available. However, you can follow the basic rules and construct these structures in assembly language or any other language. The important thing is a flexible understanding of *why* you are using structures and what characteristics make these structures useful. This is always better than a blind adherence to established rules.

One final note on structured programming. Many people confuse structured programming (an implementation technique) with the overall design strategy. These are two different processes. The two are complementary and they often are used together. However, it is completely possible to use structured implementation on any established design.

This presentation on structured programming has been, by necessity, quite brief. The principles are straightforward; putting them into real programs requires practice. It is not the easiest way to learn to program, but it offers the best long term potential for producing reliable software on time.

Debugging the Sections

You have two choices for testing and debugging your program sections. You can test them as they are developed, or you can wait and test them all at once. Either way, you want to start at the bottom and work back up. You start with the most elemental routines and check them all out. You then have an established base. You now move up and check out the routines that use the lower level ones. By moving up one level at a time you can control the amount of code being checked at one time.

One thing you never do is write five hundred lines of program, dump the whole thing into the computer and let 'er rip. You took all the time to partition and develop the program, so why blow it now? You must control your natural impatience to get the whole thing running. The best way to do it is to systematically put the thing together.

There are really two parts to program testing: deblunderizing and debugging. Deblunderizing is where you go through and correct all the spelling errors, incorrect instruction usage, syntax errors, and related detail errors. Next you move into the actual debug loop. This consists of alternately finding and correcting areas where program operation differs from the specification.

Finding the bugs is the first part. Never assume that a program that seems to run correctly has no bugs. The program should be presumed guilty until proven innocent. All large programs have bugs. Finding them requires you to devise test cases. Test cases are designed to be rotten, devious, sneaky, and otherwise adequate for finding where the system bellys up. Once you find a test case that causes an error, you trace down the responsible area and correct it. You repeat the test and debug process until you have confidence that the program section is, in fact, correct.

If you have done a good job of implementating your design, testing and debugging should be a straightforward process. A good implementation will have a relatively small number of logic errors. Logic errors are where an inaccurate algorithm causes the program to malfunction. This means the program is probably doing exactly what you told it to do. Unfortunately, you told it to do the wrong thing. The systematic design procedure is intended to catch these errors before they are committed to programs.

Final Testing and Documentation

Once you have the system completed, it will be time to turn a sample group of users loose with it for a while. This is almost guaranteed to smoke out some final bugs. Once you isolate and correct these, you are ready to document your effort. Your home system is no different than a commercial system; you need to do the documentation. Six months from now you may want to change something, and that pencil marked listing representing the sole documentation may not be very helpful.

Good documentation is usually divided into user documentation and technical documentation. User documentation consists of instructions, examples, error messages, and anything else the user will need to use the system without necessarily understanding how it really works. Technical documentation is used to help with program maintenance and modification. It consists of the original specification, flowcharts, I/O device assignments, memory usage, test cases, the latest listings, known bugs, and all other information that will be required to understand and change the program.

Summary

In this article I have hit just the high spots of a systematic approach to developing computer programs. It is a plan that can be used by all users and it has been demonstrated to produce good, reliable programs that work the way they were designed to work. It may take a little extra time to learn, but once learned it will repay the time spent many times over. ■

Clocked Logic

. . . Part 1: The

Don Lancaster
SYNERGETICS

Here it is. You software types and novices who have been looking for some good introductory material to logic elements have found the place! You say you've been looking for something on flip-flops and how they work? So have we, and it's for sure we couldn't have found a better writer to present it for you than Don Lancaster. In this first of three articles he's taken material from his upcoming book entitled CMOS Cookbook *(to be published by Howard Sams) and come up with some of the best material you'll find on the operation of the JK and D-Type flip-flops. Although he's discussing the operation of the CMOS 4027 Dual JK and the 4013 Dual D-Type, the fundamentals and operation will apply equally well to their TTL cousins, the 7473 and 7474.*

If this turns out to be an education for you, consider it a state-of-the-art one. You'll be doing your learning around a discussion of Complementary Metal-Oxide Semiconductor (CMOS) devices. With their low-power characteristics they're destined to become the next logic family to come on the scene and stir things up a little. – John.

In *clocked, synchronous* or *step-by-step* logic, the outputs of logic blocks don't change immediately after their inputs change. Instead, the logic block waits till a specific time set by a waveform on a *clock* input. Only then are output changes allowed. There are two essential steps to the clocked logic process. In the *setup* step, inputs decide what the logic block is going to do. In the *clocking* step, the logic block actually does what it was told to and provides an output.

There are lots of advantages to clocked logic. First and foremost is the orderliness of the process. Logic signals move one and only one stage at a time. This lets us move data from one block to another without unchecked races and domino effects, where a logic one or zero goes galloping several stages beyond where it was first headed. What's equally important is that a logic block's *outputs* can now determine or at least influence its own *next* output conditions without any preferential states or wild oscillations taking place.

Clocked logic also internalizes the variable processing delays from logic block to logic block. So long as the *slowest* block completes its internal operations before the next clock arrives, all outputs of all stages will be valid and predictable at the instant of clocking, so we automatically know when to look for valid data. As a side benefit, most modern clock logic is *edge sensitive*. This edge sensitivity eliminates any need for resistors and capacitors to determine a leading or a trailing edge of a logic signal.

Clocked logic is used in virtually all advanced electronic systems. This is particularly true if counting, shifting of data, or storage of characters is needed. In this article, we'll be looking first at a *do-it-yourself* clocked logic block, followed by a check into the 4013 Type D flip-flop and the 4027 JK flip-flop. These devices are extremely useful by themselves as the detailed applications catalog later in the article will show you. The same operating principles will be important as the basic building blocks when discussing the heavier counters and registers.

CMOS Clocked Logic

Most CMOS logic blocks are clocked on the *positive edge* of the clock. This is the ground to positive transition of the clock input. Clocking is defined in a positive logic sense for most CMOS devices.

There are a few exceptions to this positive clocking rule. Binary ripple counters such as

---

This article is excerpted from the *CMOS Cookbook*, copyright 1977 by Howard Sams. Reprinted by permission.

D type and JK flip-flops

the 4020, 4024, 4040, and 4060 are clocked on the negative edge or positive to ground transition of the clock. This lets you cascade binary stages for longer count lengths. A few CMOS counters give you a choice of clock polarity, set by a logic signal on a separate pin. The 4518 and 4520 dual decade and dual hexadecimal counters are the most important of these. They give you a choice of positive edge clocking for synchronous counting systems or negative edge clocking for cascaded ripple counting.

Except for a few easy-to-live-with setups and hold time limitations, *it is only the input conditions that exist at the instant of the clocking transition or edge that matter*. Inputs can change regardless of whether the clock is high or low, eliminating the *one swallowing* problems that plagued early TTL level clocked flip-flops.

There is one important clock restriction that remains with CMOS and applies to just about any logic restriction that remains with CMOS and applies to just about any logic family:

> In any clocked logic system, the clock must cycle only once, noiselessly and bounce-free, per intended output change.

This means that all our clocking signals must be clean. In particular, clocking commands that come from the outside world or from mechanical pushbuttons *must* be properly conditioned to give you one and only one clean transition per desired output change.

With CMOS, it also pays to keep the clock risetime as fast as possible. Five microseconds is a normal worst-case maximum clock transition time. If possible, make your clock signals have much faster risetimes than this. Slower risetimes may let one stage output a new state before the next stage has a chance to complete clocking. This mixes old and new data and generates garbage for you. In large CMOS systems, it pays to avoid *clock slew* problems by deriving all clock signals from the same source or from parallel sources with identical delay.

A Clocked Logic Block

One inherent feature of clocked logic is that it takes *two* regular flip-flops or storage devices to build one clocked one. One of these flip-flops takes care of accepting and setting up the input information, while the second actually carries out the intended operation and holds the result for us as an output.

In the dim distant past, one of these two storage elements consisted of a diode-capacitor memory or *steering network*. More recently, the stored charge in a base-emitter junction of a transistor served the same purpose, with presence or absence of stored charge representing a one or a zero. Today's CMOS, along with many other families, uses two distinct flip-flops — an input or *setup* storage device called the *master* and an *output* flip-flop called the *slave*. On the clocking edge, the contents of the previously setup master flip-flop is transferred to the slave. The slave flip-flop then provides us with the final output and between-clocking storage of output data.

Once again, the all important purpose of the two step process is to give us an orderly one-stage-at-a-time shift of data between clocked logic blocks and to let us count or binarily divide without getting into preferential state and hangup problems. Let's see what kind of trouble we can get into by trying to make an ordinary set-reset flip-flop binarily divide, alternating states on every input command:

In Fig. 1a is a NOR logic set-reset flip-flop. With both set and reset low, the flip-flop holds the last state it was put into, with Q and $\overline{Q}$ providing complementary outputs. A high on SET drives Q high and $\overline{Q}$ low, while a high on RESET does the opposite. Driving both SET and RESET high at the same time gives us a disallowed state, and the last input to go to ground decides the final result.

In Fig. 1b, we've converted this into sort of a clocked set-reset flip-flop. We do this by adding AND gates to the inputs, controlled by a new CLOCK line. When CLOCK is low, inputs are ignored. When CLOCK is high, inputs are accepted. We can now at least set up what the flip-flop is going to do while the clock is low and actually carry out the operation by briefly bringing the clock high.

So far, so good. This is a useful clocked logic block. We can obviously make it binarily divide by cross coupling $\overline{Q}$ to SET and Q to RESET (Fig. 1c). Now every time the clock goes high, the flip-flop will change state,

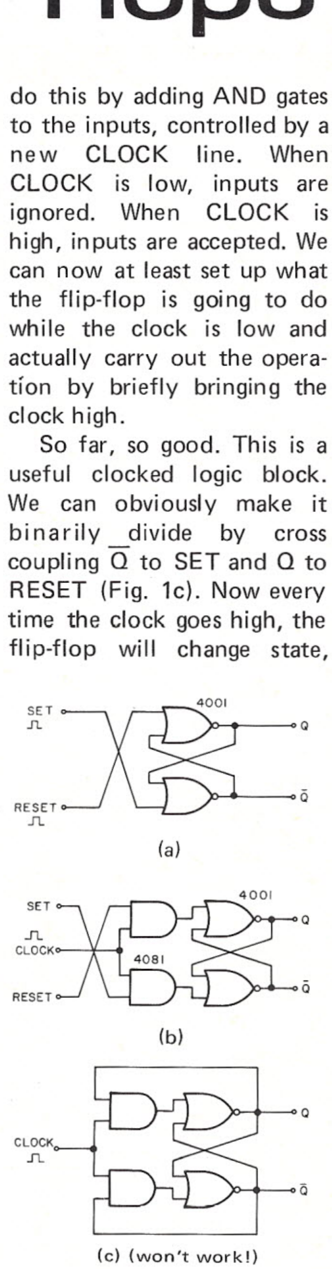


Fig. 1. Steps toward clocked logic flip-flops. (a) NOR logic set-reset flip-flop. (b) Adding AND gates gives clocking ability . . . clock input must go high to allow change of state. (c) An attempt at building a binary divider or counter that fails miserably.

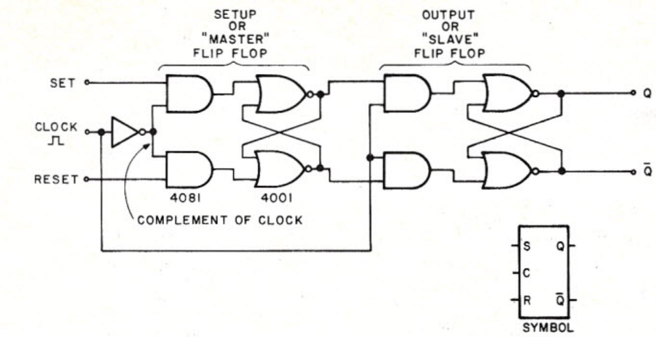


Fig. 3. Key to reliable clocked logic is the use of Master-Slave pairs of clocked flip-flops. Only one flip-flop is active at any time, eliminating unchecked races and preferential states.

since it was told to go to the opposite state. Right?

Well, not quite.

Sure enough, the instant the clock goes high, the outputs change state. But what if the clock *stays* high? These new output states reach around and change the input which changes the output which changes the input which . . . . What you really end up with is a complicated and unpredictable gated oscillator that runs while the clock is high and stops in one state or the other while the clock is low. Hardly what we had in mind.

We might try to beat the problem by picking just wide enough a clock pulse to let one and only one change take place. But this will be time, loading, device, temperature, and supply dependent. It will probably also depend on the price of yak butter futures. *The point is that there is now reliable way to let a single clocked flip-flop count or shift information*. That's why we have to use two separate storage elements or master-slave *pairs* of flip-flops for workable clocked logic.

An Alternate Action Pushbutton

Fig. 2 shows us an *alternate action* pushbutton that does work reliably. It changes its output state every time the button is pushed. At the same time, it provides free debouncing and contact conditioning.

While this circuit looks almost as simple as Fig. 1c, there is a crucial difference. Here we have two storage devices, a master capacitor and a slave flip-flop. The capacitor remembers what the new state is going to be. When the button is pressed, the capacitor voltage is transferred to the slave flip-flop. No race or oscillation is possible since the capacitor can't recharge much as long as the button is pressed, and after the button is released, no problem remains. This is a low frequency circuit ideally suited to manual button pressing.

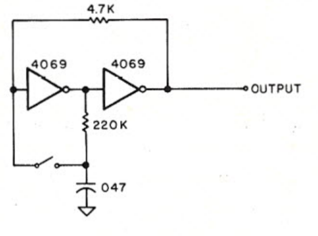


Fig. 2. Alternate action (Push on — Push off) bounceless pushbutton. Resistor and capacitor form temporary storage for "steering."

A Master-Slave Clocked Logic Block

Fig. 3 replaces the capacitor master with a conventional flip-flop. What we've done here is use two of the previous clocked NOR flip-flops. The first or master flip-flop accepts data only when the clock is low; the second or slave flip-flop only accepts data when the clock is high.

Now, when the clock is low, the master or input flip-flop can accept data and will remember the last input to go high. When the clock goes high, the input flip-flop is *disconnected* from the Set and Reset inputs and is no longer allowed to change state. But, with the clock high, the second or slave flip-flop is enabled and the contents of the master is transferred to the slave and appears as an output immediately after the clock goes high.

Even if we crosscoupled the outputs back to the inputs or cascaded stages, a wild race can't result because the next flip-flop down the line is not enabled at any particular instant.

Our circuit is said to clock on the positive edge since that's the time an output apparently appears. In reality, clocking is continuous, with the low clock state accepting data into the master and the high clock state transferring master to slave.

If we wanted a negative edge clocked flip-flop instead, we'd move the inverter so the first stage is active with clock high and the second with clock low. Note that with either system, inputs can change virtually at any time without *one swallowing* or similar problems.

We call this particular circuit a clocked RS flip-flop, and unlike our Fig. 1 circuits, its a genuinely useful building block without race or state problems. Just sitting there by itself, it can't binary divide and it still has disallowed input conditions when both Set and Reset are high, but we can fix these limitations.

It's an easy matter to convert the clocked RS flip-flop into the more useful and more common clocked logic blocks, as Fig. 4 shows us. These more common flip-flops are the type T flip-flop, the type D flip-flop, and the JK flip-flop.

The T in the T flip-flop of Fig. 4a stands for *Toggle*. By adding two external *feedback* leads from Q to reset and Q to Set, we tell the flip-flop to *change* state each time. This alternates states each clocking. Since the output changes state each positive clock transition, you only get half as many positive transitions in the output. This gives you a square wave of one half the input clocking frequency. The T flip-flop is not available separately as a CMOS package since it is easy to convert D and JK flip-flops into binary dividers. The 4024 is an example of seven cascaded T flip-flops that toggle on the negative clock edge.

A *Data* or *Delay* or Type D flip-flop is built by adding an inverter so that Reset is always the Complement of

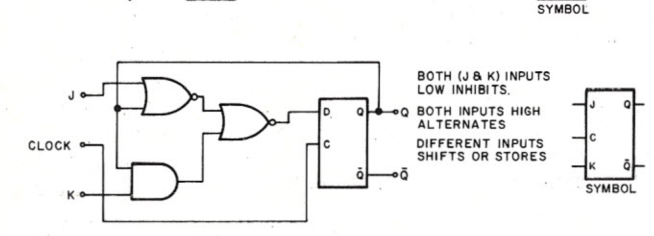


Fig. 4. Converting a clocked RS flip-flop into other clocked flip-flops.
(a) Type T flip-flop can only binary divide or alternate output states. (T represents toggle.)
(b) Type D flip-flop shifts or stores information. (D represents data or delay.)
(c) Type JK flip-flop shifts, stores, binary divides, or does nothing.

Set (Fig. 4b). A one on the D input gets stored in the flip-flop on the positive clock edge and appears at the Q output. A zero similarly applied gets clocked in and appears at the Q output. The type D flip-flop is useful in storing or delaying one bit of information. It is the key to the shift registers of the next article. We'll see that shift registers store data and move information on an orderly one-stage-at-a-time basis. We can convert a type D flip-flop to a type T flip-flop by externally feeding back the $\overline{Q}$ output to the D input.

The most versatile and universal clocked flip-flop is the JK flip-flop of Fig. 4c. The extra gates on the input make the JK flip-flop into a Type D flip-flop if the inputs are different. It makes the JK flip-flop into a Type T flip-flop if the inputs are both high. Finally, if both J and K are low, the *same* state gets reclocked back into the flip-flop making it *appear* to do nothing.

The JK flip-flop is then a universal one that can store data, binarily divide, or do nothing, all depending on the input conditions on the J and K inputs. There are no disallowed states or disallowed combinations of J and K logic. When all this versatility is needed, the JK flip-flop is the obvious choice to use, particularly for fancy or subtle timing sequences.

But the type D flip-flop is often in a shorter package, is slightly cheaper, uses somewhat less power, and often has a simpler and easier PC board layout. So, the D flip-flop is most often the best choice to use, and its a good policy to save the fancier JK versions only for those uses where you definitely need the do-nothing or inhibit option of both inputs low.

Direct Inputs

After we've gone to all the trouble of making our clocked logic block operate only when clocked and only when we want it to without any races or disallowed state conditions, we usually go back and add some new *direct* inputs that let us immediately set or reset the flip-flop into some state *independently* of the clocked inputs. We can use this to initialize a flip-flop into a certain state, to reset a group of counting flip-flops to zero, or to preset or *jam* a certain count or word into a register or latch.

These new inputs are called the Direct Set and Direct Reset inputs. Similar direct inputs on the fancier clocked logic blocks of the next article may be called Load, Preset, Reset, Clear, Jam, or some other name that suggests immediate operation independent of the clock.

> Note that all direct inputs to a clocked logic block must be disabled during clocked operation.

In CMOS, this usually means that any direct inputs are held *low* except when they are specifically used to setup, clear, or change the contents of the clocked logic blocks. Direct inputs usually dominate the clocked ones and are usually independent of the clock level or the conditions on the clocked inputs.

Generally, its a good rule to edge couple or pulse direct inputs when used — this keeps a steady direct high from hanging up your clocked logic system. When you use direct logic inputs, they always must be released before clocking.

Since the direct inputs behave as ordinary Set-Reset unclocked flip-flops, only one direct input should be used at a time. If you try using both direct inputs at once, you'll get a disallowed state condition. There is, of course, no reasonable way to let direct inputs shift or binary divide without problems — this is why we went to a clocked logic block in the first place.

The 4013 Dual D Flip-flop

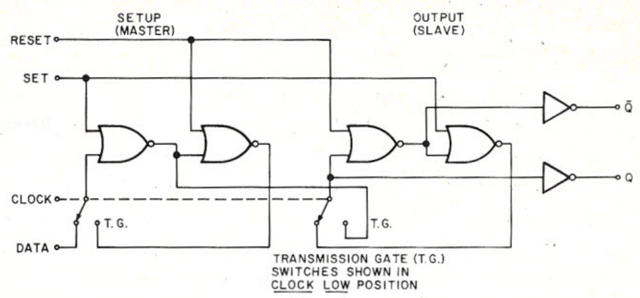


Fig. 5a. Logic diagram of half a 4013 Dual D flip-flop.

With CMOS, we can use transmission gate techniques to greatly simplify the internal design of clocked logic blocks. Let's take a detailed look at the 4013 dual D flip-flop and the 4027 dual JK flip-flops and see how they work and how transmission gates simplify the logic for us.

The logic diagram for half of a 4013 appears as Fig. 5a. While we could use AND gates for clocked logic with CMOS, the CMOS transmission gate set up as a SPDT switch greatly simplifies things for us.

Assume that the direct set and reset inputs are low. This reduces our master flip-flop to a pair of cross-coupleable inverters and does the same for the slave.

Assume further that our clock is low. The slave flip-flop is cross-coupled through its transmission gate switch, so it remembers a previous answer for us and outputs it via the buffered Q and $\overline{Q}$ outputs. These inverting buffers prevent outside loading from affecting the state or speed of operation. With the clock low, our master flip-flop is *not* cross-coupled. Instead it *follows* the data input. It will keep following the data input and remembering its instantaneous value so long as the clock is low.

As the clock suddenly goes high, the two SPDT transmission gate switches jump to the other side. This now cross-couples the master flip-flop, disconnects the master from the D input, and forces the master to remember the last value on the D input at the instant the clock went high. Since the D input goes nowhere when the clock is high, anything new to happen to the D input after the positive clock edge is ignored.

When the clock goes high, it also breaks the cross-coupling on the slave flip-flop, turning the slave into a pair of inverters that reflect the state of the master. Thus, with the clock high, the master is holding data for us and ignoring any new D inputs. The slave is simply passing on (without remembering) the master's contents directly to the outputs.

What happens when the clock goes back low? From the outside world, apparently nothing. The switches flip over to the other side. This cross-couples the slave output so it now remembers the data for us independently of what the master is up to. The master is now released and allowed to follow new input data. So, while a rather dramatic internal change takes place on the falling clock edge, no outputs can change, and things externally appear to stay as they were.

The clock rise time must be fast. Five microseconds is the usual limit. The clock must be conditioned and bounce free. A slow rise or fall time can cause switching problems where old and new data can get mixed. Note that the fall time is equally important as the rise time for proper operation. Both must be fast and clean. Note that this circuit is fully static. It can remain in the clock high or clock low states indefinitely.

We can summarize the rules for the 4013:

Both Direct inputs must be

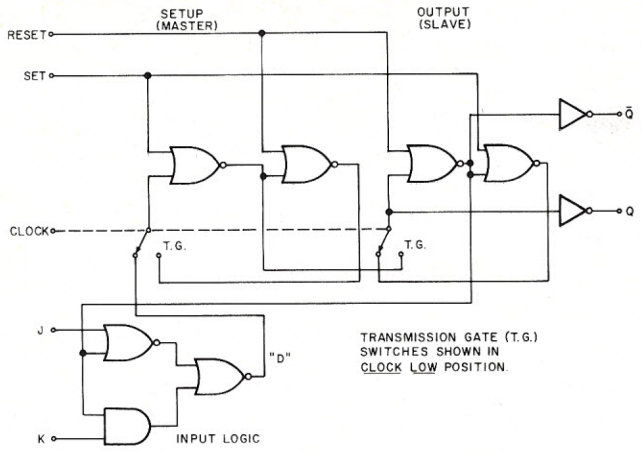


Fig. 6a. Logic diagram of half a 4027 Dual JK flip-flop.

low for normal clocked operation.

If the D input is high, the flip-flop goes or stays in the state with Q high and $\overline{Q}$ low on the positive edge of the clock.

If the D input is low, the flip-flop goes or stays in the state with Q low and $\overline{Q}$ high on the positive edge of the clock.

If the D input is cross-coupled to the $\overline{Q}$ output, the flip-flop changes to the other state on the positive edge of the clock, behaving as a binary divider.

If the Direct Set input is made high by itself, the flip-flop will immediately go or stay in the state with Q high and $\overline{Q}$ low.

If the Direct Reset input is made high by itself, the flip-flop will immediately go or stay in the state with Q low and $\overline{Q}$ high.

If the Direct Set and Direct Reset inputs are simultaneously made high, a disallowed state results with both Q and $\overline{Q}$ high, independently and dominantly over the clock and D inputs. This state is normally avoided. The last direct input to go low decides the final result.

Both direct inputs must be returned to ground before clocking can resume.

The clock must be bounceless and noise free with rise and fall times faster than five microseconds.

Fig. 5b summarizes these rules in a pair of truth tables.

The 4027 Dual JK Flip-flop

A JK flip-flop has two advantages over a type D flip-flop. We can make it binarily divide under external control and we can make it appear to do nothing (not change) despite repeated clockings. These extra performance features are obtained at the cost of a somewhat larger and more expensive IC that takes slightly more supply power in a usually more complex PC layout. The JK flip-flop is important where full performance is needed, such as in sequencers, odd-length walking ring counters, divide-by-three circuits, fully synchronous counters, and some other special uses.

The logic diagram of one half a 4027 is shown in Fig. 6a. It is the D flop circuit repeated with some funny gates added to the input. These gates respond to a J input, a K input, and an internal feedback line that monitors the *present* Q output. Since each flip-flop has one new input, we end up with a total of 16 pins, compared to the 14 of the dual 4013.

Suppose both J and K are low when we bring the clock from the low to the high state. What happens? The low K input disables the AND gate, holding its output low. The low J input is ignored by the NOR gate, and the present Q output is inverted twice and presented to point D. On clocking, the old state of the flip-flop gets *reentered.* To the outside world it looks like nothing happens at all. If J and K are both low, clock commands appear to be ignored.

What happens if J and K are both high? This will disable the NOR gate and enable the AND gate. The Q output gets inverted once and sent to D. Clocking will change the flip-flop to the other state. We alternate states or binarily divide when J and K are both high.

If J is high and K is low, the AND gate is disabled and a one unconditionally appears at D and is loaded. Similarly, if J is low and K is high, a zero unconditionally appears at point D. This zero results as a *don't care* condition. If Q is high, it goes through the AND gate, gets inverted once and ends up a zero. If Q is low, it goes through the NOR gate, gets inverted twice, but *still* ends up a zero. Either way, J low and K high loads a zero.

Our JK flip-flop acts like a type D flip-flop if the inputs are different. If both J and K are low, the circuit appears to ignore clock pulses. J and K high binarily divides.

We can summarize the rules for the 4027:

Both direct inputs must be low for normal clocked operation.

If J is low and K is low, no apparent output change takes place on the positive edge of the clock.

If J is high and K is low, the flip-flop goes or stays in the state with Q high and $\overline{Q}$ low on the positive edge of the clock.

If J is low and K is high, the flip-flop goes or stays in the state with Q low and $\overline{Q}$ high on the positive edge of the clock.

If J is high and K is high, the flip-flop changes output states, binarily dividing on the positive edge of the clock.

If the Direct Set input is made high by itself, the flip-flop will immediately go or stay in the state with Q high and $\overline{Q}$ low.

If the Direct Reset input is made high by itself, the flip-flop will immediately go or stay in the state with Q low and $\overline{Q}$ high.

If the Direct Set and Direct Reset inputs are simultaneously made high, a disallowed state results with both Q and $\overline{Q}$ high, independently and dominantly over the clock and D inputs. This state is normally avoided. The last direct input to go low decides the final result.

Both direct inputs must be returned to ground before clocking can resume.

The clock must be bounceless and noise free with rise and fall times faster than five microseconds.

Fig. 6b summarizes these rules in a pair of truth tables.

An easy way to remember the operation of the direct inputs is that if you do nothing to them (keep them low), they do nothing. On the D flip-flop, the D input gets passed across the flip-flop to the Q output on clocking. The same thing happens to the JK flip-flop with different J and K inputs. Do nothing to J and K (keep them low) and it does nothing. Do everything to J and K (both high), and you get a binary divider. ■

CLOCKED INPUTS:

D	CLOCK	Q	$\overline{Q}$
0	$\uparrow$	0	1
1	$\uparrow$	1	0
$\overline{Q}$	$\uparrow$	CHANGES	

DIRECT INPUTS:

R	S	Q	$\overline{Q}$
0	0	CLOCKED OPERATION	
0	1	1	0
1	0	0	1
1	1	1	1 (DISALLOWED)

(DIRECT INPUTS MUST BE LOW FOR CLOCKED OPERATION.)

Fig. 5b. Truth tables for 4013.

CLOCKED INPUTS:

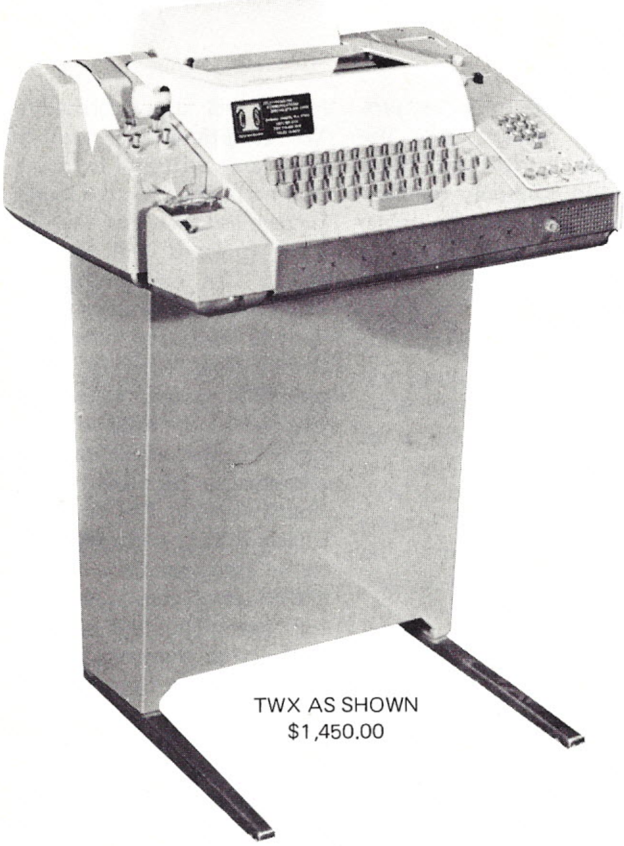
K	J	CLOCK	Q	$\overline{Q}$
0	0	$\uparrow$	NO CHANGE	
0	1	$\uparrow$	1	0
1	0	$\uparrow$	0	1
1	1	$\uparrow$	CHANGES	

DIRECT INPUTS:

R	S	Q	$\overline{Q}$
0	0	CLOCKED OPERATION	
0	1	1	0
1	0	0	1
1	1	1	1 (DISALLOWED)

Fig. 6b. Truth tables for 4027.

TELETYPE MODEL 33



TWX AS SHOWN
$1,450.00

TWX or COMPUTER INTERFACE

$840.00

- 33ASR PRIVATE-LINE
- FRICTION FEED
- COPYHOLDER & STAND
- ANSWERBACK
- MANUAL READER
- GUARANTEED 30 DAYS
- F.O.B. NEW JERSEY
- CRATING INCLUDED
- NOTHING ELSE TO BUY

Options:

- AUTOMATIC READER ADD $50
- READER RUN CARD (DEC) ADD $75
- SPROCKET (PIN) FEED ADD $100
- TAPE WINDER (ELECT.) $55 - WINDUP $22
- EIA INTERFACE $110
- TAPE UNWINDER (NON-ELECT.) $33
- PAPER WINDER (ELECTRIC) $50

--- NEW FREE CATALOG AVAILABLE NOW ---

TELETYPEWRITER COMMUNICATIONS SPECIALISTS, INC.
550 SPRINGFIELD AVENUE
BERKELEY HEIGHTS, N. J. 07922

PHONE - 201-464-5310 TWX - 710-986-3016 TELEX - 13-6479

BUY * SELL
SERVICE * LEASE

* OVERHAULING & MODIFICATIONS
* REPLACEMENT PARTS
* PAPER--TAPE--RIBBONS
* VIDEO TERMINALS
* DECWRITERS
* ACOUSTIC COUPLERS

The Gory Details of Cassette Storage

Peter Boyle
1337 Adams
Denver CO 80206

The $35 audio cassette recorder has become the standard medium for program storage on small microprocessor systems. Initially, using audio recorders to store programs was an afterthought. Many systems did not have any tape interface at all and those that did used it in the most trivial way, usually just copying memory out to the tape byte for byte. It was enough that the scheme worked at all. This attitude is changing rapidly. The audio cassette recorder has become a member in good standing of the class of mass storage devices, along with digital tapes and floppy disks.

By using higher speeds, controlling the stop-start switch and using multiple drives one can use audio cassette recorders for many of the tasks normally performed by more expensive devices. All that is required is that the subject of cassette software be taken seriously. This is what I propose to do.

Specifically, the topic here will be restricted to the subject of saving the contents of areas of memory on cassette tape, and later copying the tape back into memory. General cassette tape usage for other types of data is a subject which needs development, but that will have to be covered another time.

Terminology

For convenience, here are a few terms. I will call the program to handle cassette tape the *cassette handler*, even though this program is not a *device handler* in the strict sense. The handler has two parts. The part which writes out the memory content is the *saver* and the part which reads this data back in is the *loader*. The collection of data on the tape which the saver creates is one type of cassette tape *file*. A file of this type would be called a *save* or *memory image* file.

In some systems, where audio tape was originally conceived as the basic mass storage medium, there may exist a special form of the loader, called a *bootstrap loader*. The bootstrap loader is a minimal version of the full loader which is used to read in the entire handler. Since the form of a bootstrap loader should be determined by the details of the complete handler (rather than the other way around), the specifications of the bootstrap loader are best left until the handler details are clearly defined.

Most microcomputers have some form of *monitor* program. This is a program which is resident (in memory) at all times. It provides the basic user interface. Usually the cassette tape handler is part of this monitor program. Naturally, the monitor program, or perhaps only the bootstrap loader part of it, can be in read-only memory (ROM), and hence immediately available when a system is turned on.

The Problem

The cassette handling routines that come with commercial systems tend to be rather trivial. They are usually written for a very specific saving and loading task, with apparently no thought at all given to versatility.

In order to develop some thoughts on better cassette read-write software, I will review some of the software I've seen and then go into the alternatives. After that there will be some comments on the implementation details of a more correct set of cassette read and write routines.

Since I own a KIM-1 which is based on the MOS Technology 6502, I will draw some examples from the 6502. However the discussion will apply equally to other microcomputer systems.

The Digital Group System

In many ways, particularly in some hardware aspects, the Digital Group systems are a real pleasure. Their 6502 monitor program, however, is an excellent example of the limitations typical of cassette handling software. Their tape-read routine is on a ROM and it assumes that the file it loads will go into memory starting at location H0 (I will use a preceding "H" to differentiate numbers that are written in hexadecimal). The program loader further assumes that the program it loads will start at H500. This means that the ROM routine cannot be easily used to load some other part of memory. What is worse, programs have to be written with a hole at H500 so that a jump to the actual starting address can be placed there. That's not all. The timing is software loop controlled and these loops are in a 1702A PROM which runs at less than full speed. Thus this timing is difficult to duplicate in any other system or in normal full speed memory.

There are three lessons to learn from the limitations of the Digital Group monitor. First, cassette software should never assume that the bytes on tape correspond to some predetermined locations in memory. The location and number of bytes in the load should be part of the data that is saved on the tape. Second, the starting address of the loaded program should be saved on the tape. Third, whenever possible, external timing standards should be used.

The KIM-1

The KIM-1 is another interesting system. The KIM at least allows saving and restoring any area of memory, and it uses an external timer for timing, but it does not provide any self-starting mechanism at the end

In the following article Peter Boyle offers some criticisms of the cassette software supplied by a couple of the more popular manufacturers . . . and also puts forth some worthwhile ideas on how this software should be written. If you're not into developing cassette software at the moment, let me suggest you read the article and pick up on some of his techniques in memory management and software memory protection. — John.

of a load. This means that the starting address must be *stored* elsewhere, such as on a handy match book cover. Naturally in the course of events the files and their starting addresses tend to get separated.

Although the KIM cassette handler is a little slower than I would prefer, the cassette interface hardware is excellent. It is capable of 1200 baud (bits per second) rates, but the KIM ROM runs it at 133 baud. As soon as some additional memory was added to the KIM, I found the slow speed intolerable.

There is a more important limitation in the KIM software, which is shared by most of the cassette handlers I have seen. The memory copied out to the tape in a single write must be one continuous block and correspond to sequential memory addresses. Thus the entire write is specified by only two addresses, the upper and lower limits of the area of memory to be written out. That means that in order to write out two or more sections of memory which are in different places it is necessary to either write multiple files or copy out the in-between sections as well.

For an example of why this is a severe limitation, consider a typical KIM program. It will consist of several sections which are widely separated in the address space. It starts at, say, H200 in order to begin above the stack. It uses H0 to HFF (page 0) to store constants and commonly used variables and it requires that the interrupt vectors at H17XX be correctly set up. Between page H0 and page H17 there is 6K of memory space. To write all that as a single file requires writing all the in-between pages, most of which are not even implemented memory on the KIM.

The primary lesson from the KIM monitor is that, for one reason or another, computer programs tend not to be contiguously arranged in the address space. Clearly, cassette software should not assume that they are contiguous.

Features of Cassette Handlers

Save Time Control

So far I have discussed the things that should be specified at the time the cassette tape file is written, and hence what sort of information besides the data bytes themselves should be written onto the tape. Basically, these were the starting address of the program and the load address information. In particular, we saw that it would be nice if this load address information allowed a tape file to load any word or words in memory, independent of the addresses of these words, and even if the addresses did not follow in numerical sequence.

There are a few other items that a good cassette handler would include in file, things like the name of the file, the type of file, and perhaps even the date on which it was created.

Load Time Control

At the other end of the process — when the file is read back in — it might be nice to be able to exercise some control. It's fairly obvious that we may need to override the original load specifications. Thus the loading program should provide a mechanism for loading into a different place in memory and for starting at an address other than that saved with the program. The option of loading into a location which is specified at load time is particularly important in simple systems where a cassette handler might be used to store data other than programs. This is because data other than programs can usually be relocated to another place in memory and still be meaningful. Programs, on the other hand, usually contain references to absolute addresses that would be meaningless if the program were moved to some other part of memory.

It would also be nice if, while reading, the read program scrupulously avoided modifying any words other than those it actually read in. If this precaution is taken, then one file may overlay the current memory content, essentially adding to it. This can be useful, for example, in loading a debug routine which is only required with other programs during development.

Another feature a loader should have is the ability to load only a part of the entire file — for example, only those routines that load into memory addresses from H2000 to H2200 from a program file that could load from H2000 to H2FFF. This feature is very handy for stealing subroutines written into one program for use in another.

Commands

At this point we have listed some of the desirable features of a cassette read and write program for program storage. Before considering the implementation details it would be wise to imagine the actual operation of such a system. Imagining the actual use of a program often shows up problems that more general thoughts about it do not. This is also a good time to introduce a possible set of commands for controlling the handler.

Let's assume that a device with Teletype-like capability is available, since that is an easy to understand reference point. Let's also assume that the cassette handler is on ROM so that the details of bootstrapping a particular machine can be ignored.

Loader Commands

So now we turn on the power. The cassette handler starts automatically. We set up the tape with the program we want to load, start the tape and type "R" for "read." The loader program reads the new program into the same locations we wrote it from and starts it up. The read command needs no further specifications since the load addresses and the starting address were read from the tape file itself.

The $35 audio cassette recorder has become the standard medium for program storage on small microprocessor systems.

Fine. Suppose we only wanted part of it? In that case, we could precede the "R" command with a command specifying the highest and lowest addresses that the loader can load into. This could be "Lxxxx,yyyy" (where xxxx and yyyy are hex addresses). The loader would then ignore any data from the tape that would have loaded into an address higher than the upper limit or lower than the lower limit.

Suppose we did not want to start it after loading; for example, if we had loaded only part of a program? To handle that case we could

have a separate read command that did not start after the load, but which was in other respects the same as "R", perhaps "A" for *append.*

To provide the ability to load into a different location we could have a "Bxxxx" command which sets up a *base* address for the load. This base address might function as follows. If a load address from the tape falls within the limits set up by the "L" command, and the "B" command was used, then the base address is used instead of the address given on the tape. Bytes from the tape that normally would not be loaded into an address between the limits would be ignored.

So much for the loader part of the handler. The scenario of its use seems clear and simple. Now let's turn to the saver part of the handler.

Saver Commands

Suppose that we have a program in memory that we wish to save. It would be nice to have a command analagous to the "R" command, say "W", which simply wrote out the program in memory. Obviously, however, we do have to specify what parts of memory constitute the program and hence what needs to be written out. Ignoring for now the problem of internal representation of this information, what is the easiest way to specify it? Handled carelessly, this could be a cumbersome detail. A program can consist of bits and pieces from all over the address space. Specifying each piece to be written every time a write is made could become a nuisance.

There are some ways to avoid this nuisance. Note that the contents of memory most probably came from one or more loads from other tape files. We could preserve the information specifying which locations in memory were loaded. In the default case when only the "W" command is given we can write back out exactly those bytes we read previously.

The information saved during a load can be thought of as specifying which locations in memory are of interest. Perhaps we should call it the *memory control data* (MCD), and use it as the primary means of specifying specific areas of memory.

An MCD which was automatically updated by multiple loads would take care of copy operations, slight modifications of programs, and combining two or more parts of programs into one. This would cover most situations, but not all. There is still a need for a way to specify a write which is not just a combination of previous reads. This could be implemented by a command which modifies or extends the preserved information, specifying which bytes were loaded.

We obviously must have an "M" command, say "Mxxxx,yyyy" which would *mark* the memory between address xxxx and yyyy as important. Associated with this we will need a way to reset the MCD to a null value. Perhaps "K" for *kill.*

So far there are seven commands. These are summarized as the first seven in Fig. 1. Naturally, the computer for which the cassette handler is to be written will probably already have a system monitor program. If so, the actual syntax of the commands will probably be chosen to be compatible with the existing software.

Implementation

So, to write a cassette tape handler we must write three subprograms: the loader, which executes the "R" command; the saver, which executes the "W" command; and the command decoder which interacts with the outside world via the Teletype, accepts commands, and executes them. The loader may be split into two parts, the bootstrap and the frills.

A bootstrap for a system with this type of cassette handler would consist of the basic loader subroutine and a simple routine to call the loader and then jump to the starting address of the program loaded. Probably the bootstrap would not contain any of the optional control functions, but would allow for them after the entire loader had been input.

The command decoder is quite straightforward, so we need not go into it here. The other parts will depend upon the representations, both on the tape and in memory, that we choose for the loading data. That is what we will consider next.

R	–	Read a file into memory set bit map and start it.
W	–	Write the file specified by the bit map.
A	–	Append a file. Same as R, except do not start it.
K	–	Kill (reset) the bit map.
Bxxxx	–	Force base address for the load to Hxxxx.
Lxxxx,yyyy	–	Set upper and lower load limits.
Mxxxx,yyyy	–	Mark the pages between Hxxxx and Hyyyy on the bit map.
Pxxxx,yyyy	–	Protect the pages between Hxxxx and Hyyyy.
Oxxxx,yyyy	–	Open (unprotect) the pages between Hxxxx and Hyyyy.

Fig. 1. The keyboard commands that could be used to control the handler described in the text.

Memory Segments

How can a thing like the MCD be implemented? The obvious approach would be to keep a table of memory *segments.* Each table entry would consist of two bytes for the lower address of the segment and two bytes for the upper address. This scheme has two problems. The first is that the "M" command described above is messy to implement with a segment table arrangement. This is because a new addition to the MCD would have to be checked against the existing table entries and overlaps or duplications resolved. The second problem is purely aesthetic. As my friend Paul is wont to say, "There are only three nice numbers in computing, none, one, and as many as you like." The number of memory segments allowed would be none of these, because the space we allocated for the memory segment table would be limited. However large we made it, it would still be possible to have too many memory segments. Imagine the annoyance that a "too many memory segments" error would cause!

Bit Maps

If not a memory segment table, what then? Here is a solution that I like.

Limit the memory segments that can be written to multiples of 256 bytes (1 page). For each possible page, assign a bit as a flag which, when on, indicates that the corresponding page of memory should be written out when a "W" command is given. Since there are at most 256 possible pages we need only 32 bytes to store these 256 bits. This block of 32 words I will call the *bit map* of the current memory usage.

The bit map scheme allows any conceivable segmentation arrangement and is easy to update to reflect multiple loads or added segments as

specified by the "M" command. All we have to do is set those bits which correspond to the pages which are affected.

This arrangement for the MCD is similar to one which is used in the PDP/8 operating system. They call it a *core control block.*

The section of the code of the handler which deals with the bit map requires a little thought. One way to proceed is to write a subroutine which takes a memory page number in a register, say the accumulator, and returns two things: The number of the word in the bit map which contains the corresponding bit, and a mask word with the appropriate bit set. Fig. 2 is a flowchart of a routine which performs this task. It returns the number of the word in X and the bit mask in the accumulator. Now in order to mark a page in the bit map we call the subroutine and "OR" the mask with the appropriate bit map word. To test to see if a particular page is marked we call the subroutine and "AND" the mask with the bit map word. If the "AND" gives a nonzero result, then the page was marked.

Memory Protection

In the actual implementation of a cassette handler there is a programming difficulty that will have to be dealt with. The problem is this: There are some words of memory which the cassette loader must not be allowed to overwrite. In machines where I/O is implemented as memory addresses, the most obvious locations we must protect are the I/O ports. Since the loader is reading in from the cassette port it obviously must not write into that port. The other I/O ports probably should not be overwritten either.

In any case, the entire bit map must not be overwritten because the load time bit map will be different from the write time bit map whenever the current load is an overlay to earlier loads. If the bit map were overwritten, it would be changed to the value that existed at the time of the write, and this could be incorrect.

Besides areas of memory which must not be read by the loader there are other areas which need never be saved or reread. Memory addresses corresponding to nonexistent memory, read-only memory, and the stack area are examples.

For simplicity the entire problem could just be ignored. The user would then be expected to know what was going on and never mark certain areas in the MCD bit map. This is not a very nice solution since it requires knowledge of exactly where in memory the bit map and other special things are kept.

It would be much nicer to take care of this in the handler code itself. One way to do this would be to introduce another bit map. This map would keep track of those pages of memory that are to be *protected,* that is those pages that the user wishes to automatically exclude from any read or write.

Of course, there will have to be some commands to manipulate the protection bit map, since this feature is quite general and would have applications other than the obvious one of preventing errors produced by overwriting certain memory addresses. If the protection bit map reflects the currently implemented memory, it could be useful in a variety of other programs for run-time storage allocation. By comparing the protection bit map and the MCD bit map, a program could even determine which areas of memory are currently unused. The protection bit map could also be used to specify complex partial reads of files. The possibilities seem endless.

In keeping with the other commands we could have a "P" command similar to the "M" command suggested

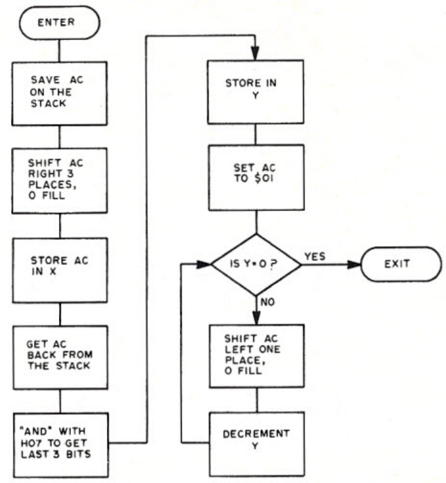


Fig. 2. This routine takes a memory page number in the accumulator and returns the corresponding bit map offset in X and the bit mask in the accumulator.

earlier. "Pxxxx,yyyy" would protect the pages between address Hxxxx and Hyyyy. Similarly "Oxxxx,yyyy" would *open*, that is unprotect, the pages between Hxxxx and Hyyyy.

Different systems and even different programs on the same system may need to protect different areas of memory. Since it is inconvenient to reset the entire protection map, it should be possible to load it from an existing file. Here is a potential problem to avoid. In general the MCD bit map will be protected while the protection bit map itself will not be protected. Since the protection scheme described so far works only on whole pages, it follows that the MCD bit map and the protection bit map will have to be stored in different pages of memory.

File Structure

At this point I would like to briefly touch on the question of the format of the file that the saver creates. The arrangement of the MCD in memory as a 32-byte block does not mean that the way to format the actual tape file is as a 32-byte MCD followed by the corresponding pages of data bytes. I made this mistake in my first version, so I know!

These are the reasons why this is a bad idea. First, it takes a bit of computing time to unpack a bit from the MCD. If the blocks on the tape are directly contiguous, there may not be enough time during the stop bit of the last character of a 256-byte page to set up for the next page. The second reason it is a bad idea is that it is an unnatural (if not impossible) way for an assembler or compiler program to format its output. The bit map scheme cannot be used unless you know, before you start the write, which pages you are going to write. Since you don't always have this information, a system standard cassette file format should not require that you do.

The format to use instead is something of an open question, and there is much to be said on the subject. Perhaps the simplest format is one used by most paper tape loaders: some number of records where each record consists of an address to begin the load at, a byte count, the actual bytes, and then a check byte or word. This format is not all encompassing, but, whatever the detail of the format, the MCD should be decoded into the actual page addresses and these actual addresses written onto the tape instead of the bit map itself. ■

Dr. Jerome Hemmye
Box 5145 College Station
Mayaquez, P.R. 00708

The Fun of Learning BASIC

... so you can write your own program

If you, or a friend, are just getting started with BASIC then let me suggest the following article as possibly some of the best introductory material you're going to run across. It's another case of a Ph.D. doing some neat writing for the world of the non Ph.D. — John.

The ability to tell your computer what you want it to do is something of an art not unlike the art of conversation. Most serious computer users sooner or later realize that they can not rely on the programs of others if they want to satisfy their own special needs. The following discussion is aimed at the group who have reached their adolescence in the world of computers. It is possible that those who are older and wiser may also find something of interest or value in what follows.

While it is true that the act of reading alone may have some value, you must not expect to gain any reasonable proficiency with the computer if you do not exercise and practice. Those who have access to a computer and actually use it will find this discussion much more worthwhile than those who do not yet fall in the ranks of the computer freaks. In these pages you will be guided along the path from the most elementary concepts in programming to the point where you are something of an expert. It is not difficult, although you will at times become deeply frustrated by your lack of ability to do what you want to do just as we are all frustrated when we don't have the precise tool we need at our finger tips. Like physical exercise, it is when it starts to hurt a bit that it is doing you good.

At the outset you should recognize that the act of programming is not really different from things you have done many times before. Without recognizing it, every time you have solved an algebraic expression you have followed a program or routine. The thing you now need to do is to convert the unrecognized ability you already have into something which will cause the computer to function in more or less the same way you make your brain perform.

As in conversation, in communicating with the computer you must have something to say, or you and the computer will never get together on anything. Unless you have a problem to solve or some type of computer activity to perform, you can't even take the first step. As we progress, we will select some arbitrary kinds of things for the computer to do. Note that while the sample programs we develop are expressed in the language called BASIC the identical concepts will apply regardless of the specific language your computer understands. We will use BASIC because it is a pretty useful language and yet it is easy enough to learn so that attention can be directed toward the concepts of programming rather than at the grammar of the language. Even machine language requires the same basic concepts which you will be learning although the grammar and construction must be handled pretty carefully or a lot of time is going to be lost debugging.

Where to start? It is well to realize that although there are a whole lot of different things going on inside that cabinet, the computer is really a pretty stupid device. Internally all it can really do is add and subtract. If you do not at this moment know how this is accomplished, don't worry, as far as the programmer is concerned it is not particularly important. You can learn all about it whenever you want — some other time. What actually goes on inside the box doesn't really make any difference to us as long as we know the rules. Anyway, the simple fact is that the computer does by addition or subtraction all the different kinds of things we care to ask it to do, and, not only that, it does them in very short order. The fact is that unless what we want done can take advantage of the speed at which the computer does its thing, there is little point in going to the computer with the problem. With this in mind, let's go to our first example.

Suppose we have a series of three numbers we wish to add up and that we have to do this a great many times. Since we must do the job many times it is a task suitable for our number cruncher, the computer. For the present we will skip the things we have to do to let the computer know we want its attention and direct ourselves toward the program steps needed to do the specific job. Conceptually, if we have three numbers in front of us, we would probably write them in some sensible order so that we could work with them. To add, we would generally write the numbers down one below the next so

that the decimal on each is properly aligned. Under the last number we would draw a line, and then go through the operation of going up or down the right-hand column adding as we go. Fortunately the computer doesn't have to be programmed at that low a level. Since we are expecting the computer to do the job many times, we will want to use some symbols for the numbers we will put in and some identity established for the result or sum. Symbolically what we want to say is A + B + C = D. Our computer can swallow all of that in one bite (no pun intended). In BASIC, we assign a step number to the operation so that the computer will know what comes first since we will need to make some additional statements in the final program. In FORTRAN and certain other languages the statement number is not always needed. BASIC also requires that we write the program step in a different order than the one you saw above. If we decide to call this step 65, we would write it as follows:

```
65 LET D=A+B+C
```

It is as simple as that. We are not really done yet since we have not provided for a way for the computer to tell us what the answer D is, nor a way for us to get the series of numbers into the computer. With a Teletype* or TV terminal, getting an answer out is simple, all we need do is tell the computer to print our results. Again we need a statement number, larger than the one for the previous step. We are going to use one somewhat larger since we may want to do some other steps in between.

```
80 PRINT D
```

* Registered trademark

Every time the computer sees the instruction to print as it is written above, it will put the last computed value for D into the printer and then automatically provide a line feed. This means that for every set of three numbers the machine operates upon, it will print a result on a line at the left, and then move the paper up a notch and print the next value of D.

> As a user however, it is always sensible to ask the question as to whose time is more valuable, yours or the computer's.

Many computer people are very concerned with how efficiently the computer is used, and would be delighted if a program step could be eliminated, and, truthfully, that is a worthwhile thing to do. As a user however, it is always sensible to ask the question as to whose time is more valuable, yours or the computer's. Program a refinement is sometimes a fun thing, so let's reduce the program we have written to half its former size:

```
65 PRINT (A+B+C)
80
```

The computer will now add and print the series of numbers all in one step. By using the statement number 65 a second time we have automatically erased what was in that statement area before and have replaced it with the new statement. By typing the number 80 with nothing after it we have erased both the statement and the statement number.

Suppose that we want to print more than one sum on a line, can it be done? The answer is yes; there are two easy alternatives. If we use a comma after the statement,

```
65 PRINT (A+B+C),
```

the page is automatically divided into five sections and the individual sum for each A, B, and C will fall into successive zones.

On the other hand,

```
65 PRINT (A+B+C);
```

will pack the numbers as close as possible across the page without causing confusion.

Incidentally, while we are thinking of the print statement, BASIC limits the printout to six significant digits, the trailing zeros are suppressed. This does lead to irregular columns when using the print statement followed by a semicolon.

Now before losing the thread of how we would solve the problem stated earlier, let's return to the program. We have seen how we may solve the algebraic expression D=A+B+C, but have not found a way to get the numerical values for the variables into the computer. Actually, there are several methods which may be used, the one you select depends on how you want to manage the data. The first method involves a DATA statement where the set of data is written right into the program. A simple example with three values, one for A, B, and C is shown in Program A. It is usually located near the end of the program, so we will number the statement accordingly.

read in sequence, that is;

```
60 READ A,B,C
```

will read A=35.69374, B=.246537, and C=10000.62. Having read the data as instructed, the computer will proceed to the next step, in this case our previous statement 65 and will print the sum A+B+C. Before going to the next statement, let's consider another way of putting data into the machine.

The INPUT statement tells the computer that you are going to enter some data from the keyboard while the program is running:

```
60 INPUT A,B,C
```

We have substituted a new statement for old number 60. When the computer encounters the INPUT statement it halts and prints a question mark. You must type in on the same line the data you want entered, followed by a carriage return. If you fail to enter data for A, B, and C, the computer will send another question mark and wait for you to enter the data and the carriage return. Be careful to enter the data in the order which you specified in your program. In this case it makes little difference but in another situation, order could be very important.

```
150 DATA 35.69374, .246537, 10000.62
```

Program A

Several observations should be made concerning the way or format in which the numbers are written. First, the comma is used to separate the individual groups of numbers and not to distinguish between hundreds and thousands, etc. Note also that while the printout is limited to six significant digits, you can use more or less than this number when you are entering data. At a later point we will show you another way to express data.

It takes an additional statement to tell the computer that you want it to read the data. Statement 150 is written so that when we read the data the first piece read will be the value assigned to the first variable. The data is

Anyhow, when you have entered all the data to satisfy the INPUT statement and have sent the carriage return, the computer will go to the next step.

Going back to the premise that the writing of a program is very much like the way you would do the same thing in your head, let's review what we have done so far. First we have inputted some data. Doing the work yourself you would have looked at and read the numbers and written them down in a column. Then the program said — print the result. You would mentally add and write down the answer under the column of figures if you were doing the work yourself. You can see that there is a great deal

```
55 FOR I=1 TO 25
60 READ A,B,C
65 PRINT (A+B+C)
150 NEXT I
200 DATA 35.69374, .246537, 10000.62, 93.728, .06328477, 100305, 83,
210 DATA 53.92854, .000314
```

Program B

of similarity between the program and what you would do if you were solving the problem yourself.

So far we have programmed only one solution, we need to make some additions to the program to make the computer repeat the solution as many times as necessary. This requires two statements:

```
55 FOR I= 1 TO 25
```

and

```
150 NEXT I
```

Statement 55 sets up a counter (called I in our case, it could have been any letter we had not previously used) and informs the computer that you expect the results of 25 different sets of data. Statement 150 has erased the old DATA statement which we are not using at the moment and tells the computer that the loop for I ends at that point and so the operation must move back to the FOR statement. Our program so far looks like this:

```
55 FOR I=1 TO 25
60 INPUT A,B,C
65 PRINT (A+B+C)
150 NEXT I
```

It doesn't matter in what order we entered the program steps in BASIC, they are sorted in a software device called a compiler or interpreter. We are presuming the compiler is available. If you are using another language, you may have to take care to maintain the statements in logical order.

What we have now is a loop in which the same operation, in this case the addition of three numbers, is done over and over until it has been done 25 times. Each time the question mark appears, you enter the data from the keyboard, and the computer prints the sum. The next example involves the data statement. The program is shown in Program B.

Here we have entered into the program nine values, enough for only three sums. Our computer would solve for three values of D, look for more data and finding none would write you an error message such as:

```
OUT OF DATA IN LINE 60
```

and would then terminate the computation. If we had only three sets of data to operate on we should change statement 55 to read:

```
55 FOR I=1 TO 3
```

and the computer would again be happy with us.

There are some other algebraic operations which we will want to do sooner or later. The plus sign (+) is the symbol for addition, and the equal sign (=) is also clear by now. As you might expect, in BASIC the minus sign (-) is used for subtraction, and division is indicated by a /. Multiplication however is different; the asterisk (*) is used instead of the familiar times sign (x) because that letter is so commonly used in equations as a variable. With these new operators we are ready for more elaborate program writing.

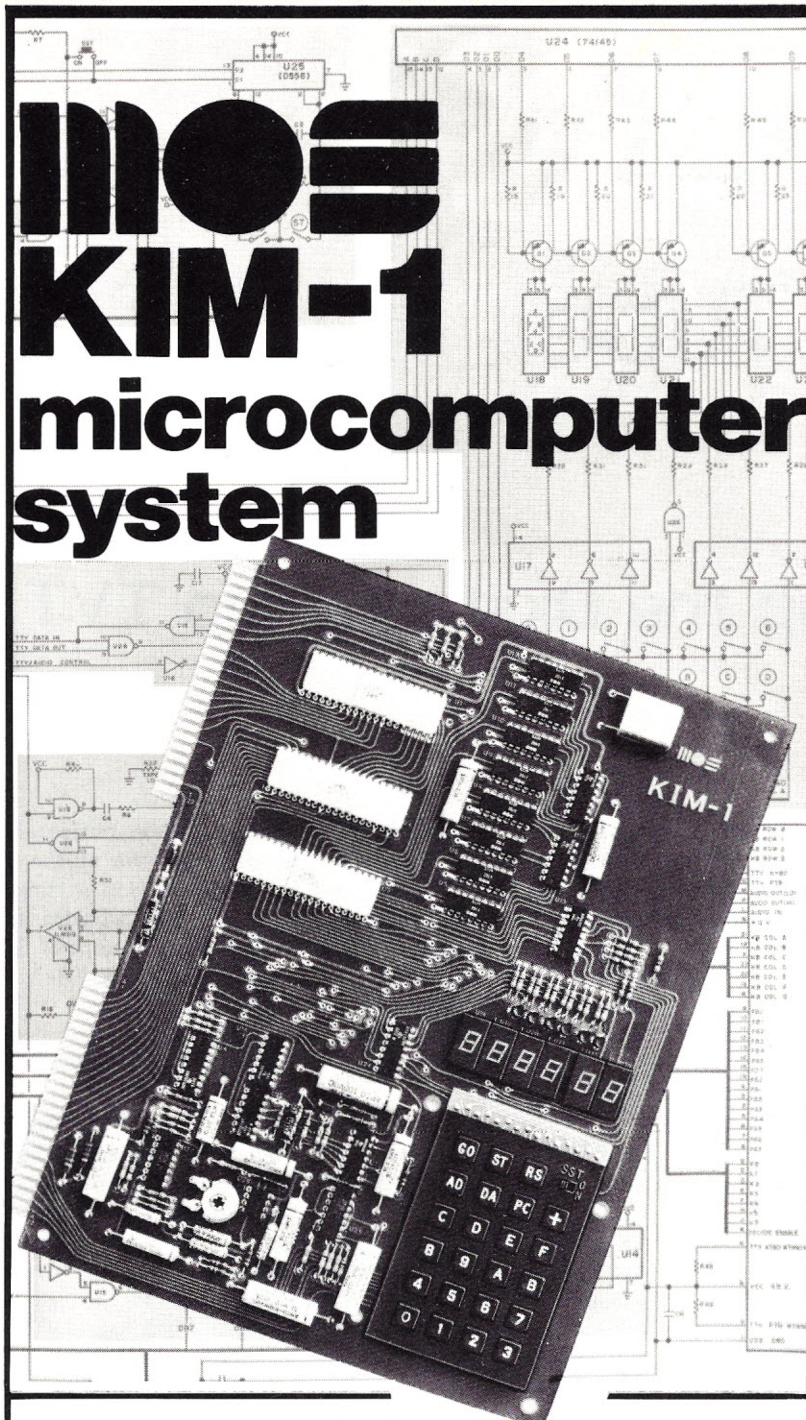
One final statement is necessary, the one which tells the computer that you have finished what you planned. The statement is:

```
250 END
```

If the END statement does not appear, the program will not run, and you will get an error statement.

In this discussion, you have learned how to name a program, perform some algebraic operations as well as how to loop. You now know how to enter data and about the end statement. Perhaps you also see that programming is quite similar to describing the way your own mind works. It is now time for you to do some exercising. You might try not only addition, but subtraction addition combinations, and then move ahead to division and to multiplication. ■

MOS KIM-1 microcomputer system



- A COMPLETE MICROCOMPUTER
- ONLY $245
- NOT A KIT!
  - FULLY ASSEMBLED
  - FULLY TESTED
  - FULLY WARRANTED
- OPERATES WITH
  - KEYBOARD & DISPLAY
  - AUDIO CASSETTE
  - TTY
- KIM-1 INCLUDES
  - HARDWARE
    KIM-1 MODULE WITH
    6502 μP ARRAY
    6530 ARRAY (2)
    1 K BYTE RAM
    15 I/O PINS
  - SOFTWARE
    MONITOR PROGRAMS
    (STORED IN
    2048 ROM BYTES)
  - FULL DOCUMENTATION
    KIM-1 USER MANUAL
    SYSTEM SCHEMATIC
    6500 HARDWARE MANUAL
    6500 PROGRAMMING MANUAL
    6500 PROGRAMMER'S REFERENCE CARD

	PRICE	SHIPPING	AVAILABILITY
KIM-1 MICROCOMPUTER	$450.00	$4.50	STOCK
KIM-2 4K MEMORY EXPANSION MODULE	179.00	3.00	STOCK
KIM-3 8K MEMORY EXPANSION MODULE	289.00	3.00	STOCK
KIM-4 MOTHERBOARD	119.00	3.00	1—31—77
DATA 1-K SOFTWARE ASSEMBLER/ EDITOR (PAPER TAPE OR CASSETTE)	250.00	2.00	STOCK

(FOR MORE THAN ONE ITEM: ONLY HIGHEST SHIPPING CHARGE APPLIES)
OHIO USERS ADD 4.5% OHIO SALES TAX.

COMING SOON: KIM-5 ROM BOARD & KIMATH ROM FLOATING POINT MATH PACKAGE.
JOHNSON COMPUTER IS ALSO STOCKING DISTRIBUTOR FOR THE OSI CHALLENGER AND THE 400 SERIES KITS.

SEND FOR OUR FREE PUBLICATION, THE uP, WHICH HIGHLIGHTS THE PRODUCTS WHICH WE HANDLE

P. O. BOX 523 MEDINA, OHIO 44256 TELEPHONE (216) 725-4560

JOHNSON COMPUTER

E. H. Sommerfield
49 Spring Road
Poughkeepsie NY 12601

Super-Tube

... jazzing up the Digital Group TVT

This article describes how a basic commercially available microprocessor kit can be enhanced with features normally found on more sophisticated minicomputer systems. The following enhancement features, although designed for "The Digital Group" TV display[1], can be applied with appropriate modifications to any similarly designed 7-bit ASCII TV display.

1. Full screen editing with characters entered, changed, and viewed on the screen.
2. Any character string can be read back into main memory from character storage.
3. Ability to display a 16-character graphic set.
4. Normal typewriter RETURN function.
5. Cursor movement up or down to any row without altering column location.
6. Cursor can be moved left or right to any column without changing the row location.
7. Ability to darken screen and/or cursor under program control.
8. Ability to read or write screen 0 (addresses 0-511) or screen 1 (addresses 512-1023) under program control; allows graphics on screen 1 to be viewed simultaneously with characters on screen 0.
9. Ability to start loading the character memory with any character and then, while awaiting END LOAD interrupt, let the CPU execute another task. When the loading is complete (about 16 ms), the character memory is reset to address 0 and the CPU is INTERRUPTED.

Before proceeding, the reader should understand the method by which the modifications are made. No printed circuit patterns have to be cut, nor are any connections made directly to any ICs. Whenever mention is made to *cutting* a connection to an IC pin, the cutting is done by inserting another *low profile* socket between the installed board IC socket and the IC, as shown in Fig. 1. The pin connection that is to be cut, is bent outwards before the extra socket is installed. Connections that are required are then made to this bent-out pin and to the now *open* land pattern.

The format of these articles is directed primarily toward *How To Do It* rather than the intimate design details that are always required to determine *Why*.

Note that there are places in the logic that appear as *inefficient* or *cumbersome* design; I'm aware of this. They came about due to the fact that as new features were added no attempt was made to rewire old logic to achieve optimized design.

Cursor Generation

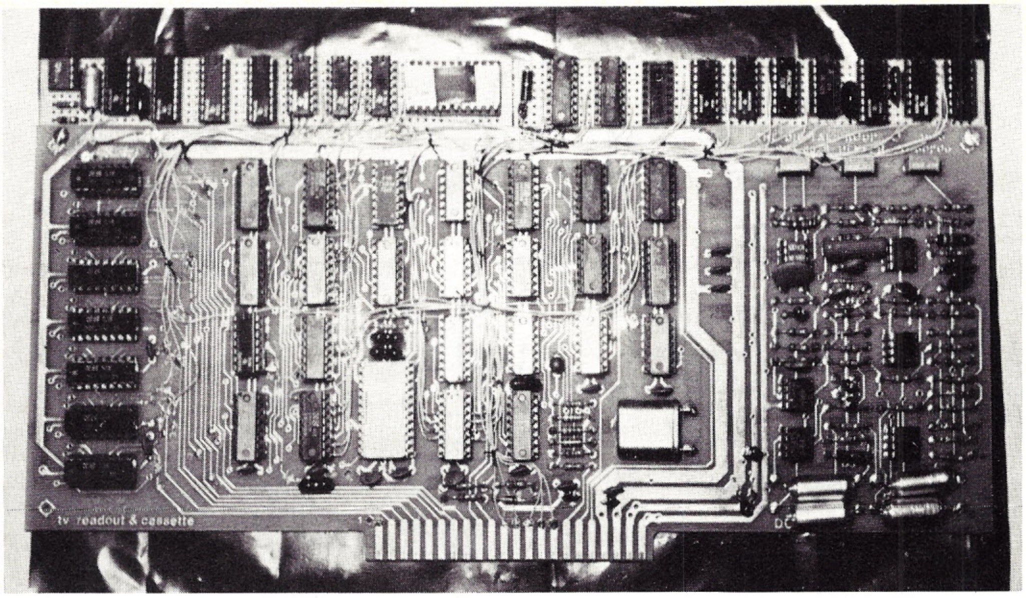
A cursor is an indication of the position of the next character location to be changed on the screen. The change may be either to replace an existing character with a blank or to enter a new character. The cursor is illuminated and blanked under control of the MPU. The cursor is generated by essentially comparing the current scanning display address with the current character memory address and when they are equal generating a cursor pulse to brighten the screen. The address equal comparison is accomplished via three 4-bit comparators — ICC, ICD, and ICE. The trailing edge of the equal compare, ANDed with the bottom line and the cursor blinker via ICA, is applied to the cursor generator SS, ICB. This one-character delay is required to place the cursor under the next character to be changed. The cursor will be illuminated, in a blinking mode, if the program controlled blinker control is a logical 1 at pin *9*. (Refer to Fig. 2.)

Nondestructive Cursor Movement

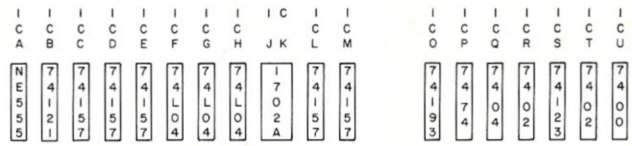
The original logic connection of the character address registers was count-up only; the count-down connection (pin 4 of IC16) was connected to +5. The count signal applied to pin 5 changed from logical 0 to 1 to 0 during strobe time.

Address change, before character storage time (character storage is delayed 600us by IC23 pin 13) is affected on the leading edge (0 to 1) for the strobe pulse. In order to move the cursor both left and right, both count-up and count-down must be done. According to the module specs, the count direction that is not being executed must be at a logical 1. Since, as mentioned above, the current count-up signal goes from 0 to 1 to 0, this signal must be inverted to 1 to 0 to 1 to be able to execute the count-down. When this is done, however, the address advance is executed after, not

[1] *Byte*, August 1976.



Sockets are low profile obtained from Digikey Corp. *Note:* Solder tab type must be used for intermediate sockets.



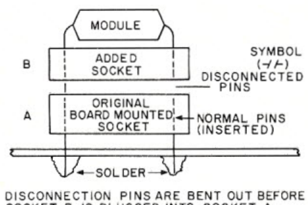


Fig. 1. Module Modification.

before, character storage. Therefore, address initialization must be to location 0, not 511, as in the original design.

This was done as shown in Fig. 3. With no cursor movements, both pins 4 and 5 are at +5. Application of either a left or right cursor movement changes the appropriate pin from a 1 to a 0 and back to a 1. Reset to location 0 will be described in the section on screen initialization.

Nondestructive UP/DOWN cursor movement, row change with no column change, is complicated by the fact that the column count (32) extends 1 bit into IC29 and requires no change, while the row count that must be changed, takes up the remainder of IC29.

During the UP/DOWN cursor movement, IC16 is not changed. Instead of incrementing/decrementing IC29, ICO keeps the count and is

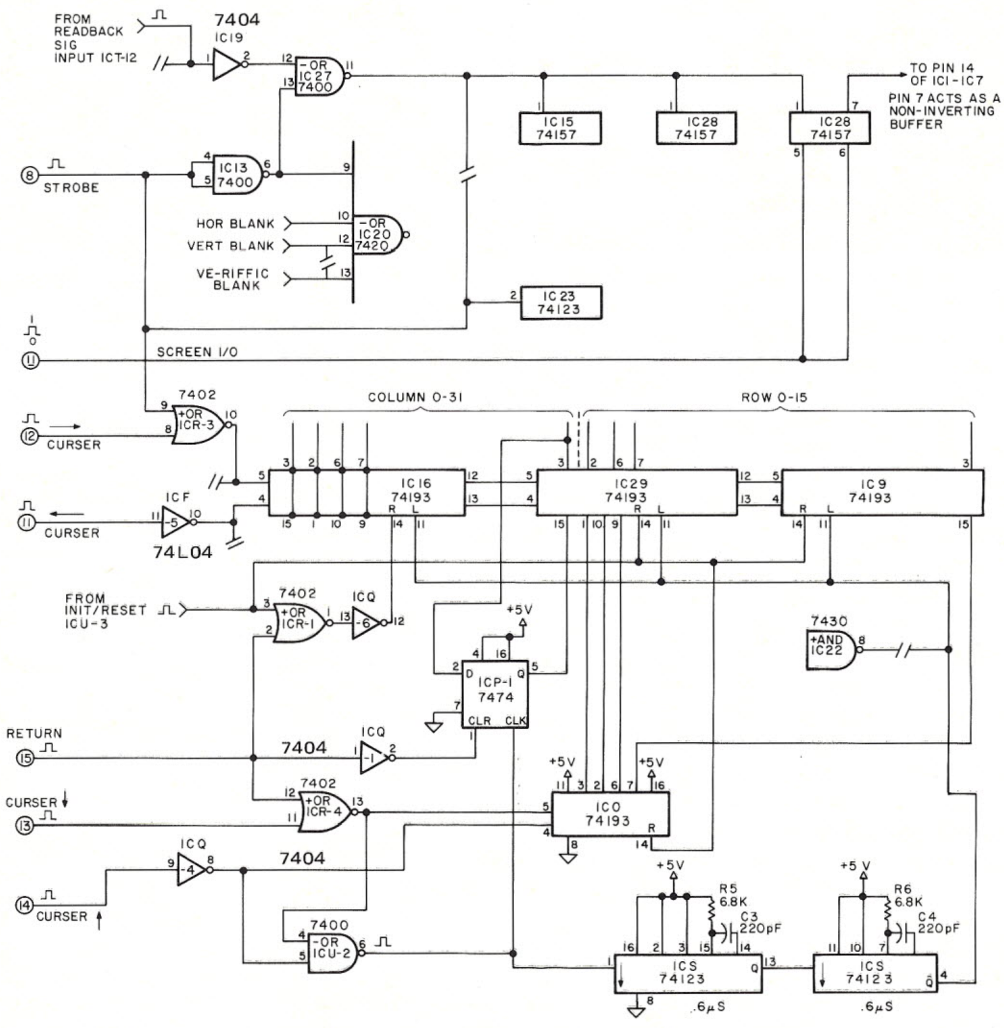


Fig. 2. Advance, cursor and blanking logic.

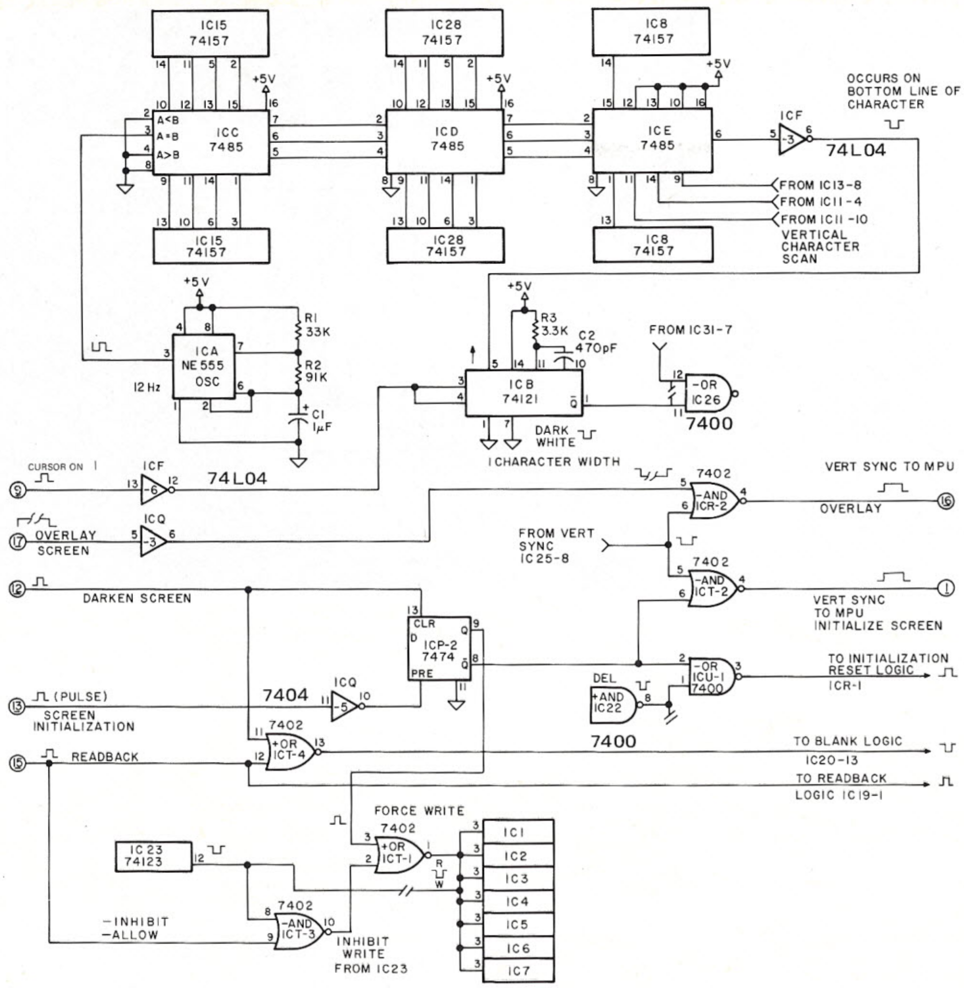


Fig. 3. Cursor generation and screen control logic.

incremented/decremented and then, 600us later, is loaded into IC29. In the absence of RETURN, ICP-5, the noninverted output is also loaded back into IC29. Therefore, IC29 bit one is not changed during UP/DOWN cursor movement.

Return is slightly different. During RETURN, the COLUMN value must be zeroed at the same time the row value must be incremented. Reset of bits 0-3 is accomplished via ICQ-11. The same loading process previously described is applied to IC29 and IC9. In this case, however, bit one of IC29 must be zeroed. This is a two-step operation. When the RETURN line is raised, ICP is reset. The leading edge of the return pulse is also applied to the clock input of ICP. Any attempt to *set* ICP by the leading edge of RETURN is overridden by the remainder of the clear pulse. Thus bit one of IC29 is always loaded as a 0 during return.

External Screen Control

This modification provides the following additional features: screen darkening, darkening of the screen during character readback, and initialization of screen to any desired character.

Screen control is applied via pin *12*. A 0 is bright and a 1 is dark. This control signal is applied via +OR ICI to IC20-13. Pin 13, separated from pin 12, provides the -OR entry to the blanking logic. (Refer to Fig. 2.)

The screen is also darkened during character readback by applying the positive readback control signal to the other input of +OR ICI.

Initialization of the screen is accomplished by a combination of software and hardware control as follows:

First the screen is darkened by a 1 applied to pin *12*: this also preconditions ICP, the initialization control BIT register. Second, a positive pulse at pin *13* sets ICP. The output of ICP:

- forces a continous *write* to the character memory, thus permitting the video scanning registers IC32, IC17, and IC10 to write the selected character into every position;
- holds the character registers IC16, IC29, and IC9 in a reset condition via ICU-3;
- Permits the 3 msec vertical sync pulses to become active on pin *1* via ICI-4. The second of such pulses to arrive at the CPU is an indication that the initialization is complete.

When initialization is complete, the darkening signal at pin *12* is removed, lighting the screen and resetting ICP.

Screen Swapping

After assembly, only 512 positions of the 1024 possible positions of the character memory are used. Unfortunately, pin 14, the MSB is connected to ground under each module socket, this forcing a 0-511 address range. In this one case only, instead of breaking the connection to pin 14 by inserting an intermediate module with pin 14 bent out, it was more expedient to break the printed circuit pattern that connects all the pin 14s to ground near IC8 pin 8 and reconnect this common lead to IC8 pin 7. Pins 6 and 5 (both A and B) inputs to IC8 are connected to socket pin *11*. This makes IC8 look like a noninverting buffer. When edge connector pin *11* is at 0, screen addresses 0-511 are enabled; when a 1, screen addresses 512-1023 are enabled. (Refer to Figs. 3 and 4.)

Readback

Readback (see Figs. 2 and 3) of character strings from the character memory ICI-IC7 is accomplished by:

- forcing IC23 into a READ-ONLY mode;
- darkening the screen so that the character being read back does not show on the screen;
- forcing the memory addressing (via IC15, IC28, and IC8) to the character address registers;
- issuing, from the MPU, a series of either left or right cursor instructions followed immediately by a read port instruction to get the character being addressed back to the MPU. *Note*: If a backwards cursor movement is desired, a single dummy back movement must be issued since the cursor is positioned to the

NEXT, rather than the current, character location.

Forced inhibit write (from IC23) is accomplished by applying a continuous positive READBACK pulse (pin *15*) to pin 9 of ICT Fig. 2. Forced character register addressing of character storage is done by applying the continuous positive readback pulse, via IC19-1 and IC27-11 (Fig. 3) to the two-way switches IC15, IC28, and IC8.

The data outputs are read from low-loading inverters ICF, ICG, and ICH. The screen is blanked, via ICT-13 to prevent the TV beam, which is continually scanning, from displaying a full screen of the readback character.

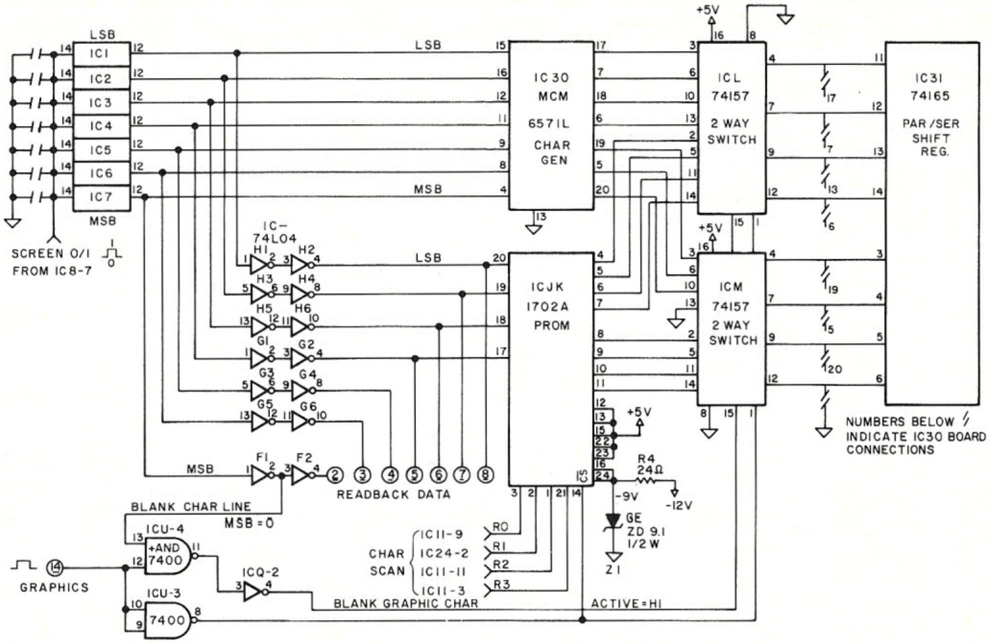


Fig. 4. Graphics generation and logic.

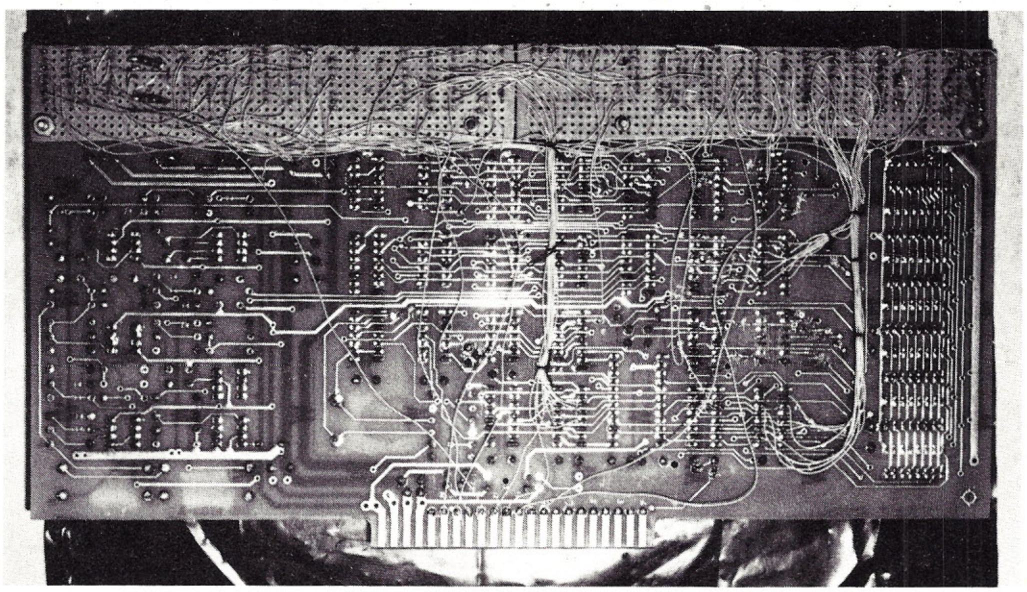
Graphics Modification

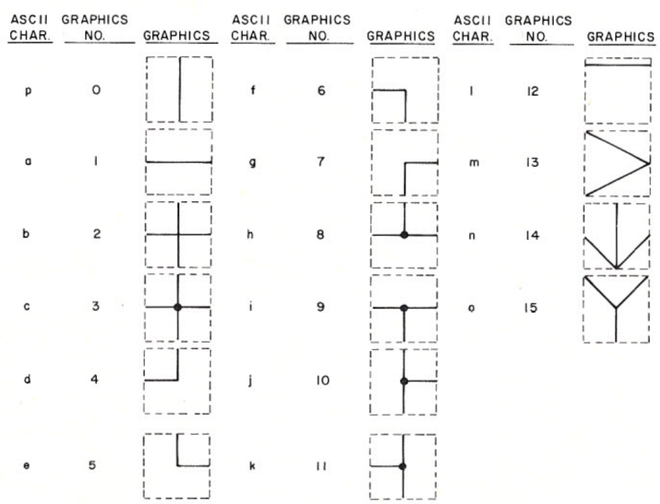
The fixed set of 16 graphics characters was implemented by paralleling the character memory to character generator data/address bus with a 1702A PROM graphic character generator ICJK. The graphics character set is detailed in Fig. 5 and Table 2. Since all 16 graphics characters were used in the PROM, a *blank* character was generated by using the MSB from the character memory IC1-IC7 via ICU-4. When MSB=0, a blank graphics character will be generated. Refer to Table 3 for an ASCII character set. Selection of which generator feeds the parallel to serial converter (IC31) is done by a signal applied to ICU via pin *14.* For minimization of power consumption, the 1702A is only gated on during graphics selection. (Refer to Fig. 4.)

Since only four of the seven output bits/characters are required, the other three can be used for color or *gray* scale control.

Construction

The 19 modules were mounted, as seen in the photograph, on 2 pieces of 1.125" x 6" perforated board, 0.100 x 0.100 C-C holes, attached to the top of The Digital Group TV display





Note: A blank can be obtained by using any character in the first 4 columns (MSB=0) of the character set shown in Table 2.

Only lower case entries are shown in Fig. 5 as they are easiest to enter. Refer to the ASCII chart for equivalents.

Fig. 5. Graphics characters.

The first digit of the address (in hex) is the graphics number (note 10-15 decimal = A-F hexadecimal).

Note*: the pattern is coded backwards with -F being the first address scanned during display of any character.*

HEX ADDR	HEX DATA	HEX ADDR	HEX DATA	HEX ADDR	HEX DATA	HEX ADDR	HEX DATA
00	00	40	00	80	00	C0	00
01	00	41	00	81	00	C1	00
02	00	42	00	82	00	C2	00
03	10	43	00	83	00	C3	00
04	10	44	00	84	00	C4	00
05	10	45	00	85	00	C5	00
06	10	46	00	86	00	C6	00
07	10	47	00	87	00	C7	00
08	10	48	00	88	00	C8	00
09	10	49	F0	89	FF	C9	00
0A	10	4A	10	8A	38	CA	00
0B	10	4B	10	8B	10	CB	00
0C	10	4C	10	8C	10	CC	00
0D	10	4D	10	8D	10	CD	00
0E	10	4E	10	8E	10	CE	00
0F	10	4F	10	8F	10	CF	FF
10	00	50	00	90	00	D0	00
11	00	51	00	91	00	D1	00
12	00	52	00	92	00	D2	00
13	00	53	00	93	10	D3	40
14	00	54	00	94	10	D4	20
15	00	55	00	95	10	D5	10
16	00	56	00	96	10	D6	08
17	00	57	00	97	10	D7	04
18	00	58	00	98	38	D8	02
19	FF	59	1F	99	FF	D9	01
1A	00	5A	10	9A	00	DA	02
1B	00	5B	10	9B	00	DB	04
1C	00	5C	10	9C	00	DC	08
1D	00	5D	10	9D	00	DD	10
1E	00	5E	10	9E	00	DE	20
1F	00	5F	10	9F	00	DF	40
20	00	60	00	A0	00	E0	00
21	00	61	00	A1	00	E1	00
22	00	62	00	A2	00	E2	00
23	10	63	10	A3	10	E3	10
24	10	64	10	A4	10	E4	38
25	10	65	10	A5	10	E5	54
26	10	66	10	A6	10	E6	92
27	10	67	10	A7	10	E7	10
28	10	68	10	A8	18	E8	10
29	FF	69	F0	A9	1F	E9	10
2A	10	6A	00	AA	18	EA	10
2B	10	6B	00	AB	10	EB	10
2C	10	6C	00	AC	10	EC	10
2D	10	6D	00	AD	10	ED	10
2E	10	6E	00	AE	10	EE	10
2F	10	6F	00	AF	10	EE	10
30	00	70	00	B0	00	F0	00
31	00	71	00	B1	00	F1	00
32	00	72	00	B2	00	F2	00
33	10	73	00	B3	10	F3	10
34	10	74	00	B4	10	F4	10
35	10	75	00	B5	10	F5	10
36	10	76	00	B6	10	F6	10
37	10	77	00	B7	10	F7	10
38	38	78	00	B8	30	F8	10
39	FF	79	1F	B9	F0	F9	10
3A	38	7A	00	BA	30	FA	10
3B	10	7B	00	BB	10	FB	10
3C	10	7C	00	BC	10	FC	10
3D	10	7D	00	BD	10	FD	28
3E	10	7E	00	BE	10	FE	44
3F	10	7F	00	BF	10	FF	82

Table 2. 1702A PROM Code for Graphics

card; connection was by handwiring. There are two places where the 4-40 flat head bolts pass through the ground bus. In order to keep ground bus continuity, the bolt heads were soldered to the ground bus.

There is one place where a signal bus must be scraped away and a jumper placed around the opening, clearing the perforated boards.

Cost

The cost for the parts is shown in Table 4. Total not counting PROM programming* — roughly $40.

Conclusion

The enhancements described in this article open the door to sophisticated applications of microprocessor video displays never before possible. A follow-on article will describe one such application now under development.■

* There are advertisements in Kilobaud and 73 for PROM programming services.

Bit Positions				HEX	0	1	2	3	4	5	6	7
			7		0	0	0	0	1	1	1	1
			6		0	0	1	1	0	0	1	1
			5		0	1	0	1	0	1	0	1
4	**3**	**2**	**1**	**HEX**								
0	0	0	0	0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	(
1	1	0	0	C	FF	FS	'	<	L		1	|
1	1	0	1	D	CR	GS	-	=	M	]	m	)
1	1	1	0	E	SO	RS	.	>	N		n	~
1	1	1	1	F	SI	US	/	?	O	—	o	DEL

MSB 7 6 5 4 3 2 1 LSB

Table 3. United States of America Standard Code for Information Exchange (USASCII).

17 Modules (average $1.00)	$17.00
26 Sockets (average $ .20)	5.20
1 1702A (average $7.50)	7.50
1 1702A Socket	.60
Miscellaneous discrete components	5.00
	$35.30

Table 4. Approximate cost for parts.

POSITIVE PULSE SIGNALS	PIN
Cursor Left	11
Cursor Right	12
Cursor Up	13
Cursor Down	14
Return	15
Screen Initialization	*13*

POSITIVE STEADY STATE LEVELS	PIN
Cursor on	*9*
Screen 0/1 Control	*11*
Darken Screen/Reset Initialization	*12*
Graphics	*14*
Readback	*15*
Overlay Screen Control	*17*

SUMMARY OF ATTENTION SIGNALS FROM TV DISPLAY	PIN
Overlay (bottom of screen)	*16*
Screen Initialization (bottom of screen)	*1*

Note: For efficiency of programming, pulse and steady state signals should be on separate CPU output groups.

Table 1 — Summary of Control Signals to TV Display

Glossary

Doug Hogg

CONTROL CHARACTER: A control character is an ASCII character whose purpose is machine control. These codes are used to pass certain commands to the computer. A control C (denoted by ↑C) is generally used to return a machine to the monitor; a line feed is a control character for an output device. In the binary representation, the control characters all have bits 6 and 7 equal to zero. All of the alphabet, punctuation marks, and numbers have bit 6, bit 7, or both, equal to one. The most common of the 35 control codes are given below.

CHARACTER	KEY	FUNCTION
BEL	↑G	bell
BS	↑H	backspace
CR	↑M or CR	carriage return
DEL	DEL	delete
DLE	↑P	data link escape
EOT	↑D	end of transmission
ESC	ESC	escape
LF	↑J or LF	line feed
NUL	NUL	null

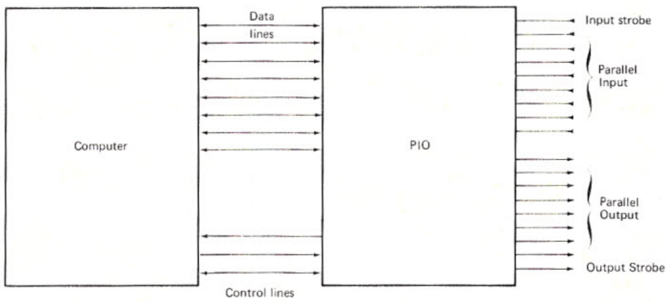
EAROM: Abbreviation for Electrically Alterable Read Only Memory. This is a type of memory which is midway between EROM (erasable read only memory) and RAM (random access memory.) EAROMs are nonvolatile (like the ROM) but are electrically erasable (EROMs are erasable only with short wavelength ultraviolet light) and can be written into with standard logic levels (most EROMs require special equipment and cannot be programmed in circuit). The main difference in the write cycle between EAROMs and RAMs is time. While typical 2102 memories have a write cycle of about 500 nsec, an EAROM write cycle may be as much as 2 microseconds and in some EAROMs the whole chip must be erased before writing. Also, there is a finite number of (about one million) write-erase cycles before the device will no longer write.

EXECUTIVE: The main program within an operating system. An executive may contain an assembler, an editor, input, output, and transfer routines, and sometimes a higher level language such as BASIC. Also called a monitor.

MONITOR: See executive.

NONVOLATILE: In reference to memories, nonvolatile refers to the ability of the memory to retain its contents when the power supplies are turned off. Volatile memory must be reloaded when the power is applied. Examples of nonvolatile memory are: ROM, magnetic cores, magnetic tape, and floppy disks. The common semiconductor RAMs are volatile.

PIO: Parallel Input-Output interface. A device which allows the computer to input and output parallel data to and from an external parallel device such as a keyboard and TVT. Parallel means that all of the data bits are output at the same time as shown in the diagram below:

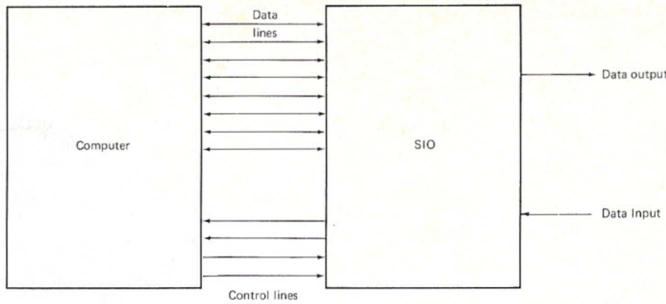


SIGNAL NOTATION: In electronics, it is convenient to give signals labels such as ENABLE. In some cases a circuit may be enabled when ENABLE is high and in other cases when it is low. In order to distinguish between the two cases, we use a

signal notation which is generally one of the forms given below. All of the terms in the left-hand column are equivalent to each other and all of the terms in the right-hand column are equivalent to each other. In use, a label from the left-hand column is employed with the corresponding label in the right-hand column.

ACTIVE HIGH	ACTIVE LOW
ENABLE+	ENABLE-
ENABLE	ENABLE
ENABLE	DISABLE
ENABLE	NENABLE
ENABLE	ENABLE'
ENABLEH	ENABLEL
ENABLE	*ENABLE

SIO: Serial Input-Output interface. The serial interface accepts data from an input device (typically a TTY keyboard or a cassette recorder) in serial form and converts this data stream into 8-bit parallel words. The interface also converts 8-bit parallel words from the computer into a serial data stream to output to a serial device such as a TTY printer. Each serial word, both input and output, consists of a start bit, 8 data bits, in some cases a parity bit, and one or two stop bits (to tell the receiver a full character has been transmitted) for a total of 10-12 serialized bits per word. The serial output circuit generates this format and the serial input circuit expects to see it. A TTY transmits at 110 baud (bits per second) and has 1 start bit, 8 data bits, and 2 stop bits. Thus, there are 11 bits per character and the character receiving and transmitting rate is 110/11 which equals 10 characters per second. (See also PIO.)



TRANSPARENT: In computer usage, transparent refers to a function not seen or directly implemented by the operator. For instance, to use a Baudot TTY with a program which expects ASCII input requires code conversion. This can be done in a routine which the operator does not see, hence the term transparent.

TRUTH TABLES: A truth table is a graphic method of expressing the possible input and output states of a logical element such as gates and flip-flops. For instance, the truth table shown here indicates that the output is high (1) only if input A *and* input B are high (1). Therefore, this is a truth table for an AND gate.

INPUTS A	INPUTS B	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

In more complicated systems the truth table for the system can be written and then used to decide which logic elements are necessary to implement the system.

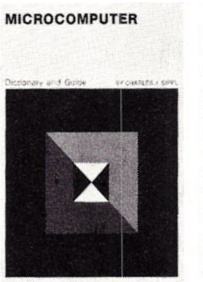
KB BOOK NOOK

COMPONENT TESTERS

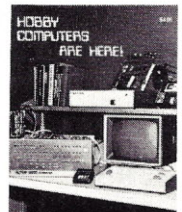
Build your own test equipment and save a bundle (and have a lot of fun). Volume 1 of the 73 Test Equipment Library shows you how to build and use transistor testers (8 of 'em), three diodes testers, 3 IC testers, 9 voltmeters and VTVMs, 8 ohmmeter, 3 inductance meters, 9 capacity meters, and a raft of other gadgets for checking temperature, crystals, Q, etc. $4.95

MICROCOMPUTER DICTIONARY

Over 5000 definitions and explanations of terms and concepts (704 pages) relating to microprocessors, microcomputers and microcontrollers. There are also separate appendices on: programmable calculators; math and statistics definitions; flowchart symbols and techniques; binary number systems and switching theory; symbol charts and tables; summaries of BASIC FORTRAN and APL. In addition there is a comprehensive electronics/computer abbreviations and acronyms section. $15.95



HOBBY COMPUTERS ARE HERE



If you (or a friend) want to come up to speed on how computers work . . . hardware and software . . . this is an excellent book. It starts with the fundamentals and explains the circuits, the basics of programming, along with a couple TVT construction projects, ASCII-Baudot, etc. This book has the highest recommendations as a teaching aid for newcomers. $4.95

RADIO FREQUENCY TESTERS

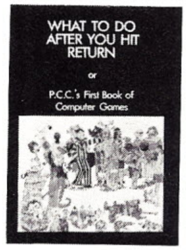
This is of more interest to hams and CBers . . . test equipment you can build for checking out transmitters and receivers: signal generators, noise generators, crystal calibrators, GDOs, dummy loads . . . things like that. This is Volume 3 of the 73 Test Equipment Library (Prepublication offer) $4.95

AUDIO FREQUENCY TESTERS

If you're into audio . . . such as digital cassette recording, RTTY, Baudot vs ASCII, SSTV, SSB, Touchtone or even hi-fi . . . you'll want to have this book full of home built test equipment projects. Volume II (Prepublication offer) $4.95

What To Do After You Hit Return

PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, muchly illustrated. Lunar landing, Hammurabi, King, Civil 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $6.95 pp.



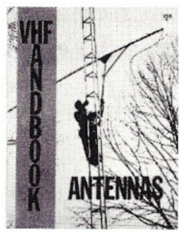
THE NEW COMPUTERS

This book takes over where the previous book leaves off. This, like the other, is a collection of reprints from recent issues of 73 Magazine (you've been missing a lot of very valuable data). This is one of the easiest ways to really understand how micros work and how to use them. Written entirely by hobbyists (prepublication offer). $4.95

VHF ANTENNA HANDBOOK

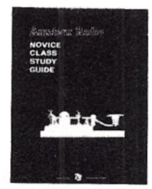
The NEW VHF Antenna Handbook details the theory, design and construction of hundreds of different VHF and UHF antennas . . .

A practical book written for the average amateur who takes joy in building, not full of complex formulas for the design engineer. Packed with fabulous antenna projects you can build. $4.95

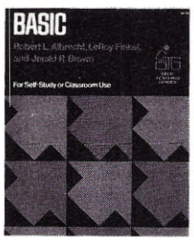


NOVICE STUDY GUIDE

This is the most complete Novice study guide available. It is brand new. This is not only invaluable for anyone wanting to get started in amateur radio, but also it is about the only really simple book on the fundamentals of electricity and electronics. And without your fundamentals down pat, how can you go on to really understand and work with computers? First things first. $4.95



BASIC . . . *by Bob Albrecht, etc.*



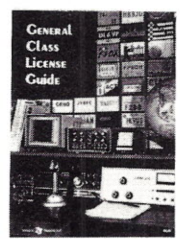
Self-teaching guide to the computer language you will need to know for use with your microcomputer. 324 pages. $4.95 pp.

Computer Programming Handbook

A complete guide to computer programming and data processing. Includes many worked out examples and history of computers. **$8.95**

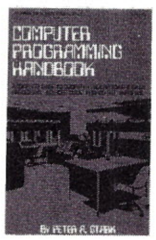
GENERAL CLASS STUDY GUIDE

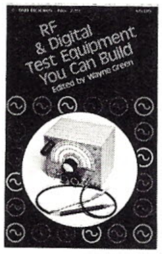
This book takes over on theory where the Novice book leaves off. You'll need to know the electronic theory in this to work with computers and you'll not find an easier place to get the information. It will also make getting your Tech or General license a breeze . . . then you can get on the ham repeaters and interconnect your micro with others. $5.95



WEATHER SATELLITE HANDBOOK

Simple equipment and methods for getting good pictures from the weather satellite. Antennas, receivers, monitors, facsimile you can build, tracking, automatic control (you don't even have to be home). Dr. Taggart WB8DQT $4.95.





RF AND DIGITAL TEST EQUIPMENT YOU CAN BUILD

RF burst, function, square wave generators, variable length pulse generators — 100 kHz marker, i-f and rf sweep generators, audio osc, af/rf signal injector, 146 MHz synthesizer, digital readouts for counters, several counters, prescaler, microwavemeter, etc. 252 pages. $5.95.

101 GAMES IN BASIC

Okay so once you get your computer up and running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you, from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. $7.50 pp.

SSTV HANDBOOK

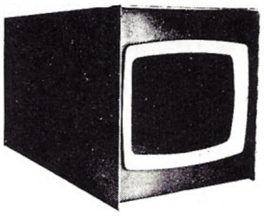
This excellent book tells all about it, from its history and basics to the present state of the art techniques. Contains chapters on circuits, monitors, cameras, color SSTV, test equipment and much more.

Hardbound $7 Softbound $5

TTL COOKBOOK

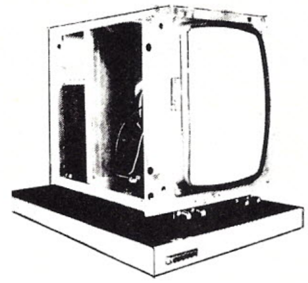
by Donald Lancaster. Explains what TTL is, how it works, and how to use it. Discusses practical applications, such as a digital counter and display system, events counter, electronic stopwatch, digital voltmeter, and a digital tachometer. 336 pages; 5½ x 8½; softbound. **$8.95**

KILOBAUD MAGAZINE ● PETERBOROUGH NH 03458 ● USE ORDER CARD INSIDE BACK COVER!
● TOLL FREE SUBSCRIPTION # (800) 258-5473

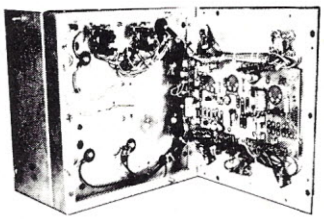


Encl.+ Bezel $7.95

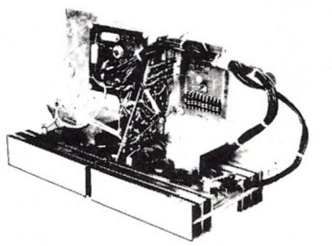
ASCII KBD $39.95



Chassis w/CRT $29.95



LVPwr. Sply. $29.95



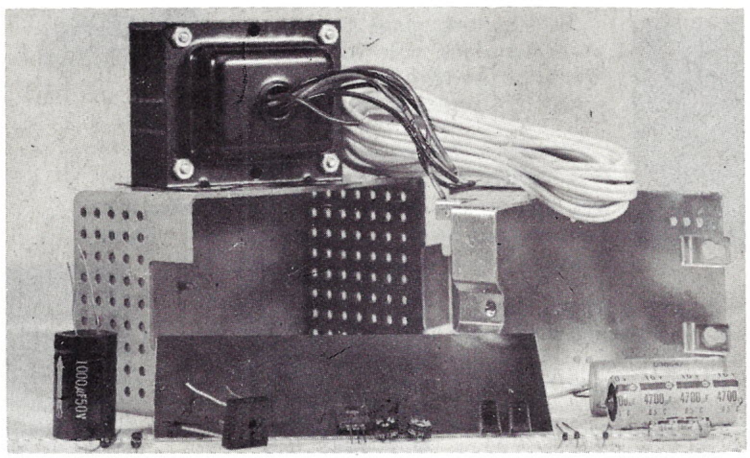
H&V Defl. Amp $39.95 pr.

HVPS 12kv $29.95

Separate prices total $177.70 - Buy all for only **$130** + shipping.
Checked out OK 6 mos. ago, but not guaranteed at this price.
Order now & we'll include 4 PC Cards free!! DATA Incl.

HOBBYIST POWER SUPPLY KITS - NEW FOR 1977

ALWAYS OVERLOOKED WHEN PLANNING A SYSTEM OR PROJECT. ALWAYS ONE MORE NEEDED FOR THE BENCH. ALWAYS NEED A LITTLE MORE CURRENT OR A DIFFERENT VOLTAGE - ALWAYS A PROBLEM. BUT NO LONGER - SUNTRONIX HAS DESIGNED A VERY VERSATILE GROUP OF POWER SUPPLY KITS WHICH CAN BE CONSTRUCTED BY THE HOBBYIST IN LITTLE TIME. ONLY AVERAGE TOOLS AND SKILLS ARE REQUIRED BECAUSE WE'LL TELL YOU HOW AND PROVIDE ALL THE PARTS NEEDED. WITH THESE BRAND NEW PARTS AND OUR INSTRUCTIONS YOU WILL NEVER AGAIN HAVE TO FACE THE PROBLEM OF THE MISSING POWER SUPPLY. ALL KITS ARE SUPPLIED WITH ALL OF THE PARTS NEEDED TO CONSTRUCT THE POWER SUPPLY THAT WILL SUPPLY THE VOLTAGE AND CURRENT SELECTED BY YOU FROM THE CHART. LOCATE THE VOLTAGE AND CURRENT RANGE YOU WANT AND ORDER BY THAT NUMBER. WE'LL DO THE REST. EACH KIT WILL PROVIDE ANY VOLTAGE WITHIN THE LISTED RANGE AT ANY OUTPUT CURRENT UP TO THE MAXIMUM AMPS LISTED IN THE CURRENT ROW. FOR EXAMPLE, KIT NUMBER 2720 WILL PROVIDE UP TO 2.0 AMPS AT ANY VOLTAGE YOU SPECIFY WITHIN THE RANGE OF 2.0 TO 7.0 WITH MINIMUM RIPPLE AND A REGULATION PRECISION OF ±0.05% AT LINE VOLTAGES FROM 95 TO 120 VAC AND LOAD CURRENTS FROM MINIMUM TO MAXIMUM. NOT BAD FOR THE PRICE, HUH?



CURRENT \ VOLTAGE	2.0 TO 7.0	7.0 TO 12.0	12 TO 20	20 TO 35
UP TO 0.150A	#2715 $ 9.95	71215 $ 10.95	12215 $12.95	20415 $14.95
0.150 TO 2A	2720 $16.95	71220 $ 19.95	12202 $24.95	20402 $28.95
2A TO 5A	2750 $24.95	71250 $ 29.95	12205 $34.95	20405 $39.95
5A TO 10A	27100 $34.95	71210 $ 39.95	12210 $44.95	20410 $49.95

Terms: Full price plus shipping cost must accompany order. No CODs. All prices subject to change without notice.

SUNTRONIX company

your BANKAMERICARD welcome here

360 Merrimack Street, Lawrence MA 01843 617-688-0751
Hours: 8:00 am to 5:00 pm

master charge THE INTERBANK CARD

S.D. SALES CO.

P.O. BOX 28810 - K
DALLAS, TEXAS 75228

Z–80 CPU CARD KIT FOR IMSAI/ALTAIR

$149.$ KIT

From the same people who brought you the $89.95 4K RAM kit. We were not the first to introduce an IMSAI/ALTAIR compatible Z—80 card, but we do feel that ours has the best design and quality at the lowest price.

The advanced features of the Z—80 such as an expanded set of 158 instructions, 8080A software compatibility, and operation from a single 5VDC supply, are all well known. What makes our card different is the extra care we took in the hardware design. The CPU card will always stop on an M1 state. We also generate TRUE SYNC on card, to insure that the rest of your system functions properly. Dynamic memory refresh and NMI are brought out for your use. Believe it or not, not all of our competitors have gone to the extra trouble of doing this.

As always, this kit includes all parts, all sockets, and complete instructions for ease of assembly. Because of our past experience with our 4K kit we suggest that you order early. All orders will be shipped on a strict first come basis. Dealers inquiries welcome on this item.

Kit shipped with 2 MHZ crystals for existing 500NS memory. Easily modified for faster RAM chips when the prices come down.

Z–80 Manual – $7.50 Separately.

Kit includes Zilog Manual and all parts.

JUMBO LED CAR CLOCK

$16.95 KIT

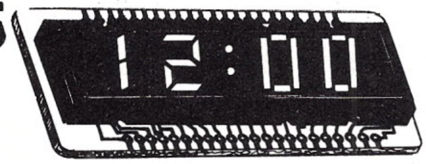
You requested it! Our first DC operated clock kit. Professionally engineered from scratch to be a DC operated clock. Not a makeshift kluge as sold by others. Features: Bowmar 4 digit .5 inch LED array, Mostek 50252 super clock chip, on board precision time base, 12 or 24 hour real time format, perfect for cars, boats, vans, etc. Kit contains PC Board and all other parts needed (except case). 50,000 satisfied clock kit customers cannot be wrong!

FOR ALARM OPTION ADD $1.50
FOR XFMR FOR AC OPERATION ADD $1.50

STICK IT! in your clock in your DVM, etc.!

Huge Special Purchase Not Factory Seconds As sold by others!

$3.95



4 JUMBO .50" DIGITS ON ONE STICK! (with colons and AM/PM Indicator)

BUY 3 for $10.

BOWMAR 4 DIGIT LED READOUT ARRAY

The Bowmar Opto-Stick. The best readout bargain we have ever offered. Has four common cathode jumbo digits with all segments and cathodes brought out. Increased versatility since any of the digits may be used independently to fit your applications. Perfect for any clock chip, especially direct drive units like 50380 or 7010. Also use in freq. counters, DVM's, etc. For 12 or 24 hour format.

60 HZ CRYSTAL TIME BASE FOR DIGITAL CLOCKS
S.D. SALES EXCLUSIVE!

KIT FEATURES:

A. 60HZ output with accuracy comparable to a digital watch.
B. Directly interfaces with all MOS Clock Chips.
C. Super low power consumption. (1.5 ma typ.)
D. Uses latest MOS 17 stage divider IC.
E. Eliminates forever the problem of AC line glitches.
F. Perfect for cars, boats, campers, or even for portable clocks at ham field days.
G. Small Size, can be used in existing enclosures.

$5.95 or 2/$10.

KIT INCLUDES CRYSTAL, DIVIDER IC, PC BOARD PLUS ALL OTHER NECESSARY PARTS & SPECS

UP YOUR COMPUTER!
21L02–1 1K LOW POWER 500 NS STATIC RAM
TIME IS OF THE ESSENCE!

And so is power. Not only are our RAM'S faster than a speeding bullet but they are now very low power. We are pleased to offer prime new 21L02–1 low power and super fast RAM's. Allows you to STRETCH your power supply farther and at the same time keep the wait light off.

8 for $12.95

$12.95 S.D. SALES EXCLUSIVE! MOS 6 DIGIT UP-DOWN COUNTER $12.95

40 PIN DIP. Everything you ever wanted in a counter chip. Features: Direct LED segment drive, single power supply (12 VDC TYP.), six decades up/down, pre-loadable counter, separate pre-loadable compare register with compare output, BCD AND seven segment outputs, internal scan oscillator, CMOS compatible, leading zero blanking. 1MHZ. count input frequency. Very limited qty.

WITH DATA SHEET

50HZ CRYSTAL TIME BASE KIT – $6.95

All the features of our 60HZ kit but has 50HZ output. For use with clock chips like the 50252 that require 50HZ to give 24 hour time format.

WESTERN DIGITAL UART

No. TR1602B. 40 pin DIP

This is a very powerful and popular part.

NEW–$6.95 with data
LIMITED QUANTITY

SLIDE SWITCH ASSORTMENT

Our best seller. Includes miniature and standard sizes, single and multi-position units. All new, first quality, name brand. Try one package and you'll reorder more. SPECIAL 12/$1.

THIS MONTH'S SPECIALS!

SPECIAL

300.00 KHZ CRYSTAL – $1.50
8080A – CPU CHIP by AMD – $19.95
82S129 – 256 x 4 PROM – $2.50
N.S. 8865 OCTAL DARLINGTON DRIVERS 3 for $1.00
Z-80 – CPU by ZILOG – $69.95
MM5204 – 4K EPROM – $7.95

Prices in effect this month ONLY!

SPECIAL

RESISTOR ASSORTMENT

¼ W 5% and 10% PC leads. A good mix of values. 200/$2.

MOTOROLA POWER DARLINGTON

Back in Stock!

Like MJ3001. NPN 80V. 10A. HFE 6000 TYP. TO–3 case. We include a free 723C volt reg. with schematic for power supply. SPECIAL–$1.99

1702A 2K ERASEABLE PROM'S – $6.95

We tell it like it is. We could have said these were factory new, but here is the straight scoop. We bought a load of new computer gear that contained a quantity of 1702 A's in sockets. We carefully removed the parts, verified their quality, and are offering them on one heck of a deal. First come, first served. Satisfaction guaranteed! U.V. Eraseable.

$6.95 each 4 for $25

4K LOW POWER RAM BOARD KIT
THE WHOLE WORKS – $89.95

Imsai and Altair 8080 plug in compatible. Uses low power static 21L02–1 500ns. RAM's, which are included. Fully buffered, drastically reduced power consumption, on board regulated, all sockets and parts included. Premium quality plated thru PC Board.

7400—19c	7411—29c	7451—19c	7490—65c	74153—75c
74LS00—49c	7413—50c	7453—19c	74LS90—95c	74154-1.00
7402—19c	7416—69c	7473—39c	7492—75c	74157—75c
74LS02—49c	7420—19c	7474—35c	7493—69c	74161—95c
7404—19c	7430—19c	74LS74-59c	7495—75c	74164-1.10
74L04—29c	7432—34c	7475—69c	7496—89c	74165-1.10
74S04—44c	7437—39c	7476—35c	74121-38c	74174—95c
74LS04-49c	7438—39c	7480—49c	74123-65c	74181-2.50
7406—29c	7440—19c	7483—95c	74132-1.70	74191-1.25
7408—19c	7447—85c	7485—95c	74S138-1.95	74192-1.25
7410—19c	7448—85c	7486—45c	74141-75c	74193-1.00
				74195—69c

TTL INTEGRATED CIRCUITS

TERMS:

Money Back Guarantee. No COD. Texas Residents add 5% tax. Add 5% of order for postage and handling. Orders under $10. add 75c. Foreign orders: US Funds ONLY!

CALL YOUR BANK AMERICARD OR MASTER CHARGE ORDER IN ON OUR CONTINENTAL UNITED STATES TOLL FREE WATTS:

1–800–527-3460

Texas Residents Call Collect

214/271-0022

S.D. SALES CO.
P.O. BOX 28810 K
Dallas, Texas 75228

For orders over $15.00 Choose $1.00 FREE mdse.

CRYSTALS

THESE FREQUENCIES ONLY

Part #	Frequency	Case/Style	Price
CY1A	1.000 MHz	HC33/U	$5.95
CY2A	2.000 MHz	HC33/U	$5.95
CY3A	4.000 MHz	HC18/U	$4.95
CY7A	5.000 MHz	HC18/U	$4.95
CY12A	10.000 MHz	HC18/U	$4.95
CY14A	14.31818 MHz	HC18U	$4.95
CY19A	18.000 MHz	HC18/U	$4.95
CY22A	20.000 MHz	HC18/U	$4.95
CY30B	32.000 MHz	HC18/U	$4.95

XR-2260KB Kit $27.95 XR-2206KA Kit $17.95

EXAR

WAVEFORM GENERATORS		MISCELLANEOUS		TIMERS	
XR-205	$8.40	XR-2211CP	$6.70	XR-555CP	$ .39
XR-2206CP	4.49	XR-4136	.99	XR-320P	1.55
XR-2207CP	3.85	XR-1468	3.85	XR-556CP	1.85
STEREO DECODERS		XR-1488	5.80	XR-2556CP	3.20
XR-1310CP	$3.20	XR-1489	4.80	XR-2240CP	3.25
XR-1310EP	3.20	XR-2208	5.20	**PHASE LOCKED LOOPS**	
XR-1800P	3.20			XR-210	5.20
XR-2567	2.99			XR-215	6.60
				XR-567CP	1.95
				XR-567CT	1.70

CONNECTORS

PRINTED CIRCUIT EDGE-CARD

.156 Spacing-Tin-Double Read-Out

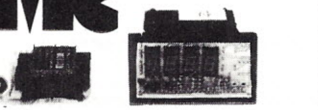
Bifurcated Contacts — Fits .054 to .070 P.C. Cards

15/30	PINS (Solder Eyelet)	$1.95
18/36	PINS (Solder Eyelet)	$2.49
22/44	PINS (Solder Eyelet)	$2.95
50/100 (.100 Spacing)	PINS (Solder Eyelet)	$6.95

25 PIN-D SUBMINIATURE

DB25	PLUG	$3.25
DB25	SOCKET	$4.95

IMC 3½ DIGIT DVM KIT



This 0-2 VDC .05 per cent digital voltmeter features the Motorola 3½ digit DVM chip set. It has a .4" LED display and operates from a single +5V power supply. The unit is provided complete with an injection molded black plastic case complete with Bezel. An optional power supply is available which fits into the same case as the 0-2V DVM allowing 117 VAC operation.

A. 0-2V DVM with Case $49.95

B. 5V Power Supply $14.95

VECTOR WIRING PENCIL

Vector Wiring Pencil P173 consists of a hand held featherweight (under one ounce) tool which is used to guide and wrap insulated wire, fed off a self-contained replaceable bobbin, onto component leads or terminals installed on pre-punched "P" Pattern Vectorbord. Connections between the wrapped wire and component leads, pads or terminals are made by soldering. Complete with 250 FT of red wire. **$9.95**

REPLACEMENT WIRE — BOBBINS FOR WIRING PENCIL

W36-3-A-Pkg. 3	250 ft. 36 AWG GREEN	$2.40
W36-3-B-Pkg. 3	250 ft. 36 AWG RED	$2.40
W36-3-C-Pkg. 3	250 ft. 36 AWG CLEAR	$2.40
W36-3-D-Pkg. 3	250 ft. 36 AWG BLUE	$2.40

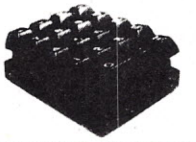
1/16 VECTOR BOARD

	0.1" Hole Spacing Part No.	P-Pattern L	W	Price 1	2-Up
PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 02XXXP	4.50	17.00	3.69	3.32
EPOXY	64P44 062	4.50	6.50	2.07	1.86
GLASS	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

HEAT SINKS

205-CB	Beryllium Copper Heat Sink with Black Finish for TO-5	$ .25
291-.36H	Aluminum Heat Sink for TO-220 Transistors & Regulators	$ .25
680-.75A	Black Anodized Aluminum	$1.60

HEXADECIMAL ENCODER 19-KEY PAD



- 1 - 0
- ABCDEF
- Return Key
- Optional Key (Period)
- — Key

$10.95 each

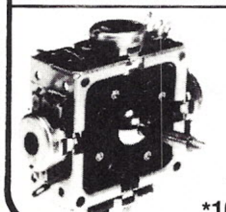
63 KEY KEYBOARD

This keyboard features 63 unencoded SPST keys, unattached to any kind of P.C.B. A very solid molded plastic 13" x 4" base suits most applications.

$19.95

HD0165 16 LINE TO FOUR BIT PARALLEL KEYBOARD ENCODER $7.95

JOYSTICK



These joysticks feature four * potentiometers, that vary resistance proportional to the angle of the stick. Sturdy metal construction with plastics components only at the movable joint. Perfect for electronic games and instrumentation.

***5K Pots $6.95**

***100K Pots $7.95**

MICROPROCESSOR COMPONENTS

8080A	CPU	$19.95	MC6800L	8 Bit MPU	$35.00
8212	8 Bit Input/Output	4.95	MC6820L	Periph. Interface Adapter	15.00
8216	Bi-Directional Bus Driver	6.95	MC6810AP1	128 x 8 Static RAM	6.00
8224	Clock Generator/Driver	10.95	MC6830L7	1024 x 8 Bit ROM	18.00
8228	System Controller - Bus Driver	10.95			

CPU'S			RAM'S			
8008	8 BIT CPU	$19.95	1101	256 x 1	Static	$ 2.25
8080	Super 8008	24.95	1103	1024 x 1	Dynamic	1.00
8080A	Super 8008	19.95	2101	256 x 4	Static	5.95
SR'S			2102	1024 x 1	Static	1.75
2504	1024 Dynamic	$ 3.95	2107	4096 x 1	Dynamic	9.95
2518	Hex 32 BIT	7.00	2111	256 x 4	Static	6.95
2519	Hex 40 BIT	4.00	7010	1024 x 1	MNOS	29.95
2524	512 Dynamic	2.49	7489	16 x 4	Static	2.49
2525	1024 Dynamic	6.00	8101	256 x 4	Static	6.95
2527	Dual 256 BIT	3.95	8111	256 x 4	Static	6.95
2529	Dual 512 BIT	4.00	8599	16 x 4	Static	3.49
2532	Quad 80 BIT	3.95	91L02	1024 x 1	Static	2.25
2533	1024 Static	7.95	74200	256 x 1	Static	6.95
3341	Fifo	6.95	93421	256 x 1	Static	2.95
74LS670	16 x 4 Reg.	3.95	MM5262	2K x 1	Dynamic	.99
UART'S			**PROMS**			
AY-5-1013	30K Baud	$5.95	1702A	2048	Famos	$12.95
ROM'S			5203	2048	Famos	14.95
2513	Char. Gen.	$ 9.95	82S23	32 x 8	Open C.	5.00
2516	Char. Gen.	10.95	82S123	32 x 8	Tristate	5.00
74S387	1024-Bit Programmable	1.95	74S287	1024	Static	7.95
			3601	256 x 4	Fast	3.95

Special Programing Available — BIPOLAR PROM SPECIAL — Write or Call for Pricing

6330-1	256 Bit (32 x 8) Open Collector	2.95	6306-1	2048 Bit (512 x 4) Three State	9.95
6331-1	256 Bit (32 x 8) Three State	2.95	6340-1	2048 Bit (512 x 8) Open Collector	19.95
6300-1	1024 Bit (256 x 4) Open Collector	3.49	6341-1	2048 Bit (512 x 8) Three State	19.95
6301-1	1024 Bit (256 x 4) Three State	3.49	6352-1	4096 Bit (102 x 4) Open Collector	19.95
6305-1	2048 Bit (512 x 4) Open Collector	9.95	6353-1	4096 Bit (1024 x 4) Three State	19.95

Continental Specialties

Proto Board 100 $19.95

Here's a low cost, big 10 IC capacity breadboard kit with all the quality of QT Sockets and the best of the Proto-Board series ... complete down to the last nut, bolt and screw. Includes 2 QT-35S Sockets, 1 QT-35B Bus Strip, 2 5-way binding posts, 4 rubber feet, screws, nuts, bolts and easy assembly instructions.

THE MINI-BREADBOARD BUDGET KIT

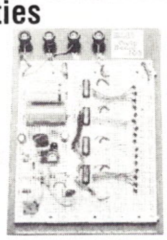
Proto Board 104. Huge 3,060 solderless tie points, (4) 5-way binding posts, 32-14 pin DIP capacity, 9½" long x 8" wide **79.95**

COMPLETE KIT ...

Proto Board 101. Compact, inexpensive, 10-14 pin DIP capacity, 5.8" long x 4.5" wide **29.95**

Proto Board 102. Compact 12-14 pin DIP capacity 7" x 8½" **39.95**

Proto Board 103. 2,250 solderless tie points, (4) 5-way binding posts, 24-14 pin DIP capacity 9" x 6" **59.95**

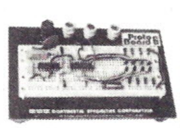


DESIGN MATE 1

CIRCUIT DESIGNER

You asked for it. HERE IT IS! CSC's basic circuit designer to help you build and test ANY electronic circuit. Solid #22 AWG wire interconnects any discreet component without solder. Resistors, Transistors, Linear/Digital ICs in TO-5/DIP packages (from 8-40 pins). And more. Components fit directly into Sockets and Bus strips instantly and easily. Plus, DESIGN MATE ONE's (DM-1) variable regulated power supply gives 5-15VDC up to 600ma (9 watts). Now that's design flexibility for you!

You can even monitor the DM-1's internal power supply or external circuits thanks to the built-in 0-15V voltmeter. If you're into electronics, DM-1 is a must for your laboratory. It comes to you completely wired, tested and ready to use for only **$54.95**



PROTO BOARD 6

The PB-6 lets the user test and built circuits without soldering or patch cords ... all interconnections between components are made with common #22 AWG hook-up wire. This quality breadboarding kit includes 630 component tie points at less than 2.5 cents each. It measures 6" long by 4" wide. Designed specially to Breadboard Microprocessor Circuits.

Proto Board 6 $15.95

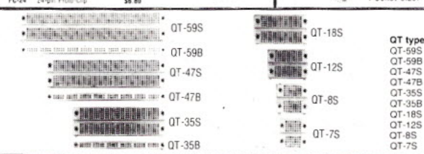
Now trace signals or troubleshoot fast. Inject signals or wire unused circuits into existing boards. Scope probes, test leads lock onto Dynagrip inset. Plastic construction — no springs or pivots. Thousands of operations. Narrow throat for high density PC boards.

PC-14	14-pin Proto-Clip	$3.60 ea.
PC-16	16-pin Proto-Clip	$3.80
PC-24	24-pin Proto-Clip	$6.80

SPECIAL!

LOGIC MONITOR

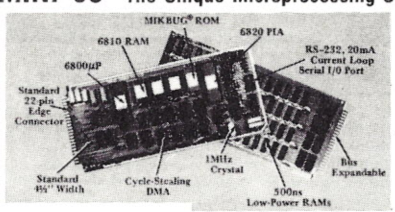
Simultaneously displays static and dynamic logic states of DTL, TTL, HTL or CMOS DIP ICs. Pocket size. $84.95.



QT Proto Strips

QT type	#holes	price
QT-59S	590	12.50
QT-59B	bus strip	2.50
QT-47S	470	10.00
QT-47B	bus strip	2.25
QT-35S	350	8.50
QT-35B	bus strip	2.00
QT-18S	180	4.75
QT-12S	120	3.75
QT-8S	80	3.25
QT-7S	70	3.00

GEMINI-68 The Unique Microprocessing System



ALL BOARDS BUS EXPANDABLE

Uses standard size 4½" wide boards, dual 22 pin edge connector

Fully buffered and tristatable address and data buses

STAND ALONE CPU BOARD — Has 384 bytes of RAM on board, serial I/O (RS-232 and 20 ma current loop, cycle stealing direct memory access (DMA), built in software — selectable echo-back capability. Part # SA-CPU Board **$279.95**

CPU BOARD — Same as above but only has 128 bytes of RAM on board-used with 8K RAM board listed below Part # Gemini 68 CPU Board **$259.95**

8K RAM BOARD — Uses low power static RAMS, 500ns cycle time, 1.5 Amps Max Part # Gemini 68 RAM Board **$269.95**

8K EPROM BOARD — Uses 5204 EPROMS by AMI or NATIONAL. Shipped with all decode and miscellaneous IC's, except the 5204 EPROMS Part # Gemini 68 EPROM Board **$ 89.95**

NOT A KIT — ALL BOARDS ARE COMPLETELY ASSEMBLED, BURNED-IN AND TESTED. COMES WITH COMPLETE DOCUMENTATION.

Allow approximately four weeks for delivery

$5.00 Minimum Order — U.S. Funds Only
California Residents — Add 6% Sales Tax

Spec Sheets - 25¢ — Send 24¢ Stamp for 1977 Catalog
Dealer Discount Available — Request Pricing

NOW OPEN SATURDAYS

James ELECTRONICS

1021-A HOWARD AVE., SAN CARLOS, CA. 94070
PHONE ORDERS WELCOME — (415) 592-8097
All Advertised Prices Good Thru March

DIP SWITCH

These switches feature seven SPST slide switches in a molded dip. They are ideally suited for microprocessor applications. **$1.95**

Timeband™*

*A Trademark of Fairchild Camera and Instrument Corporation

DIGITAL ALARM CLOCK $16.95

- 24-Hour Alarm
- "DOZE" Button
- 100% Solid State
- Large Red Led Display (.8" high)
- AM/PM Indicator
- Seconds Display at touch of button
- SPECIFY BLACK OR IVORY

DIGITAL WATCHES

Ladies Watch
- 6 Function
- Bracelet Styling
- 1 Year Guarantee
- Model 900
- Specify Gold or Chrome

$59.95

EXELAR Mens Watch
- 5 Function
- Quartz Crystal
- Black Leather Band
- Manufacturer Guarantee
- Specify Gold or Chrome

$25.00

5 FUNCTION ELECTRONIC CALCULATOR RADOFIN MODEL 8P

$8.95

FEATURES:
- 8 Digit Display
- 5 Functions consists of addition, subtraction, multiplication, division, percentage with constant on all functions, with full floating decimal point
- Power source is 1 piece 9V DC Battery 00GP, jack for AC adapter
- Black superfine grained finish plastic cabinet

DIGITAL STOPWATCH

- Bright 6 Digit LED Display
- Times to 59 minutes 59.59 seconds
- Crystal Controlled Time Base
- Three Stopwatches in One
  Times Single Event — Split & Taylor
- Size 4.5" x 2.15" x .90" (4½ ounces)
- Uses 3 Penlite Cells.

Kit — $39.95
Assembled — $49.95
Heavy Duty Carry Case $5.95

DIGITAL QUARTZ CAR CLOCK

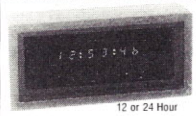
Complete kit from mounting bracket of the injection molded case down to the three conductor power cord and all components including MM5314 clock chip. Features quartz accuracy of .01%, six digit, .35" high LED display, and P.C. Boards. Works on any 12 volt system — motorcycles, boats, vans, motorhomes, autos, and trucks.

Kit: $29.95
Assembled: $39.95

DIMENSIONS: 4½" x 4" x 2"
12 or 24 HOUR MODE

CASE ONLY (includes hardware, mounting bracket and bezel) **$5.95**

JE700 CLOCK



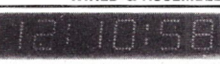
The JE700 is a low cost digital clock, but is a very high quality unit. The unit features a simulated walnut case with dimensions of 6" x 2½" x 1". It utilizes a MAN72 high brightness readout, and the MM5314 clock chip.

12 or 24 Hour

115 VAC **$17.95**

This large digit clock (.6" hours & minutes, .3" seconds) features the MM5314 clock chip. It operates from 117 VAC, and will operate in either a 12 or 24 hour mode. The clock is complete with a walnut grain case, and has fast set, slow set, and hold time set features.

JE500 KIT - ALL COMPONENTS & CASE $34.95
WIRED & ASSEMBLED $39.95



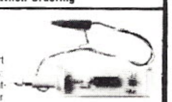
DIGITAL CLOCK KIT — 3½ INCH DIGITS

4 DIGIT KIT $49.95 **4 DIGIT ASSEMBLED $59.95**
6 DIGIT KIT $69.95 **6 DIGIT ASSEMBLED $79.95**

This clock features big 3½" high digits for viewing in offices, auditoriums, etc. Each digit is formed by 31 bright 0.2" LED's. The clock operates from 117 VAC, has either 12 or 24 hr. operation. The 6 digit version is 27" x 3½" x 1½" and the 4 digit is 18" x 3½" x 1½". Kits come complete with all components, case and transformer.

Specify 12 or 24 Hour When Ordering

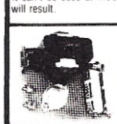
JE803 PROBE



The Logic Probe is a unit which is for the most part indespensible in trouble shooting logic families TTL, DTL, RTL, CMOS. It derives the power it needs to operate directly off of the circuit under test, drawing a scant 10 mA max. It uses a MAN3 readout to indicate any of the following states by these symbols: (H) - 1 (LOW) - 0 (PULSE) - P. The Probe can detect high frequency pulses to 45 MHz. It can't be used at MOS levels or circuit damage will result.

$9.95 Per Kit
printed circuit board

T²L 5V 1A Supply



This is a standard TTL power supply using the well known LM309K regulator IC to provide a solid 1 AMP of current at 5 volts. We try to make things easy for you by providing everything you need in one package, including the hardware for only. **$9.95 Per Kit**

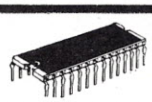
- Accuracy: ±0.05% of Reading ±1 Count
- Two Voltage Ranges: 1.999 V and 199.9 mV
- Up to 25 Conversions/s
- Z_{in} > 1000 M ohm
- Auto-Polarity and Auto-Zero
- Single Positive Voltage Reference
- Standard B-Series CMOS Outputs—Drives One Low Power Schottky Load
- Uses On-Chip System Clock, or External Clock
- Low Power Consumption: 8.0 mW typical @ ±5.0 V
- Wide Supply Range: e.g., ±4.5 V to ±8.0 V

MC14433 SINGLE CHIP 3½ DIGIT A/D
Single chip combines linear and CMOS digital to bring you the simplest yet DVM approach. Requiring only 4 external passive parts, this subsystem gives you: Auto polarity, auto zero, single voltage reference, 8 mW operation, overrange, underrange signals, 25 conversions per second and .05% ± 1 count accuracy! 100 uV resolution. 24 Pin DIP.
MC14433P...........with specs................$19.55

MC14412 UNIVERSAL MODEM CHIP
MC14412 contains a complete FSK modulator and de-modulator compatible with foreign and USA communications. (0-600 BPS)
FEATURES:
.On chip crystal oscillator
.Echo suppressor disable tone generator
.Originate and answer modes
.Simplex, half-duplex, and full duplex operation
.On chip sine wave
.Modem self test mode
.Selectable data rates: 0-200
0-300
0-600
.Single supply
VDD=4.75 to 15VDC - FL suffix
VDD=4.75 to 6 VDC - VL suffix
TYPICAL APPLICATIONS:
.Stand alone - low speed modems
.Built - in low speed modems
.Remote terminals, accoustic couplers
MC14412FL................................ $28.99
MC14412VL................................ $21.74
6 pages of data....................... .60
Crystal for the above......$4.95

MC14411 BIT RATE GENERATOR.
Single chip for generating selectable frequencies for equipment in data communications such as TTY, printers, CRT s or microprocessors. Generates 14 different standard bit rates which are multiplied under external control to 1X, 8X, 16X or 64X initial value. Operates from single +5 volt supply. MC14411....................... $11.98
4 pages of data................................ .40
Crystal for the above.......................... $4.95

REMOTE CONTROL TRANSMITTER. MC14422P is a 22 channel ultra-sonic remote control transmitter I.C. CMOS uses little power and only a few external passive components. Applications include TV receivers, security controls, toys, industrial controls and locks. 16 pin DIP plastic pkg.
MC14422P........... with specs.................$11.10



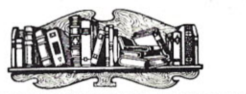
MC6525 REMOTE CONTROL RECEIVER
The MC6525 is a 22-channel remote control receiver circuit designed for use in television receivers, industrial remote controls, remote security controls, radio receivers, electronic games and similar applications. The circuit is intended for use with the MC14422 remote control transmitter. Comes in 28 pin DIP plastic package.
MC6525P........with 6 pages of specs........... $18.38

3 DECADE (BCD) COUNTER CHIP
MC14553BCP consists of 3 negative edge triggered synchronous counters, 3 quad latches and self scan multiplexed , TTL compatible outputs.
MC14553BCP.............................$8.72
Spec sheets...............................$.60

LM1889 TV VIDEO MODULATOR
The LM1889 is designed to interface audio, color difference, and luminance signals to the antenna terminals of a TV receiver. It consists of a sound subcarrier oscillator, chroma subcarrier oscillator, quadrature chroma modulators, and R.F. oscillators and modulators for two low-VHF channels.
The LM1889 allows video information from VTR's, games, test equipment, or similar sources to be displayed on black and white or color TV receivers.
LM1889 with 16 pages of data $9.95, data only, $1.00

OSCILLATOR MODULE.
Complete clock oscillator and buffer for micro computer applications. Crystal controlled at 18.432 MHZ ± .01% - exactly right for the 8080A and baud rate generator when divided by ten. These brand new units were built by Motorola at several times our price. For 5 volt operation and P.C. mounting.
K-1100A............with spec................ $13.90

MINIATURE POWER SUPPLY
5V, .25A, P.C. mounting module supplies logic voltage from 105-125VAC input, 50-400HZ. Ultra-stable and noise free with external trim capability. Made by PMC. (Model MM-5BL)..........$15.95



DATA BOOKS BY NATIONAL SEMICONDUCTOR
DIGITAL. Covers TTL, DTL, Tri-State, etc.......$3.95
LINEAR. Covers amplifiers, pre-amps, op-amps... $4.25
LINEAR APPLICATIONS VOLUME I. Dozens of application notes and technical briefs covering the use of op-amps, regulators, phase locked loops and audio amps.................................... $3.25
LINEAR APPLICATIONS VOLUME II. Takes up where Volume I left you--All the latest linear devices. Along with Vol I you have a great source of application data on the most widely used devices as well as new types just appearing.....................$3.25
CMOS Gates, Flip Flops, registers, etc........$3.00
VOLTAGE REGULATORS. A must for anyone making a power supply. Complete theory including transformers, filters, heat sinks, regulators etc........... $3.00
MEMORY. Info on MOS and Bipolar memories, RAMS
ROMS, PROMS and decoder/encoders........... $3.95
INTERFACE. Covers peripheral drivers, level translators, line driver/receivers, memory and clock drivers, sense amps, display driver and opto-couplers..... $3.95
SPECIAL FUNCTIONS DATA BOOK. Contains detailed information for specifying and applying special amplifiers, buffers, clock drivers, analog switches and D/A-A/D converter products.................. $3.25
AUDIO HANDBOOK Contains detailed discussions, including complete design particulars, covering many areas of audio with real world design examples... $3.25
-------------------SPECIAL--------------------
DATA BOOKSHELF. Buy all ten of the National Data books at one time and save $5.10!!!..........$30.00
(All books shipped ppd in US only. Foreign orders please add shipping for 1.5 lbs per book)

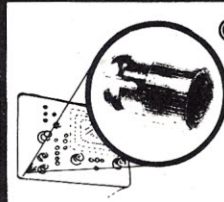
AMP L'ANNY
Says

IF YOUR PROJECT TIME AND MONEY ARE BEING BLOWN AWAY BY DELAYS AND HIGH PRICES---
****** MARCH ON DOWN TO TRI-TEK! ********

HOBBY-WRAP TOOL
Battery operated with built-in 30 ga bit and sleeve. Uses standard C batteries (not included). Light weight, only 11 oz. Wraps standard DIP sockets. Has built in device to prevent overwrapping. Pistol grip, positive indexing. Quality construction assures exceptional performance. BW-630....................$34.95
FREE, a 50' roll of wire wrap wire with each tool!!!

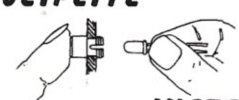
HOBBY-WRAP-30 Handy little tool to strip, wrap and unwrap 30 gauge wire......$5.95

CLIPLITE™
COMBINATION LENS AND MOUNTING DEVICE FOR T 1¾ LED



REQUIRES NO TOOLS

SNAP CLIPLITE



INSERT LED

AVAILABLE IN TRANSPARENT RED - GREEN - AMBER - CLEAR & YELLOW

CLIPLITE
Combination lens and mounting device for T 1-3/4 LED. The CLIPLITE combines the benefits of the present LED display panel mounting methods and eliminates their deficiencies. Requires no special tools and installs in 6 seconds in .250" hole. Simple two-step installation. Just snap CLIPLITE, insert LED. Available in transparent red, green, amber, clear and yellow. Specify colors, any mix.
5/$1.00, 10/$1.90, 20/$3.50, 50/$7.50, 100/$13.50

78H12 HIGH CURRENT REGULATOR
Now, a 12V, 5A regulator in a single TO-3 package!! Internal short circuit limit and thermal overload protection. Same ease of application as the popular 309K
78H12...............with specs............. $14.25

INTRODUCTION TO MICRO COMPUTERS
New book from OSBORNE.
The first edition of this classic was a huge success. Now, due to the growth of information on the subject Osborne has expanded the work into 2 volumes. Vol I covers basic concepts, Vol II discusses real world micro computers.
IMC-002 Vol I...................................$8.00
IMC-002 Vol II.................................. $13.00

'NOTHER NEW BOOK FROM OSBORNE.
"8080 PROGRAMMING FOR LOGIC DESIGN" explains how an assembly language program within a microcomputer system can replace combinatorial logic ---- for logic designers, programmers or anyone who is interested in real and powerful applications of the ubiquitous 8080.
PLD-4001...................................... $8.00

tri-tek, inc.
6522 north 43rd avenue,
glendale, arizona 85301
phone 602 - 931-6949

We pay surface shipping on all orders over $10 US, $15 foreign in US funds. Please add extra for first class or air mail. Excess will be refunded. Orders under $10, add $1 handling. Please add 50¢ insurance. Master charge and Bank America cards welcome, ($20 minimum). Telephone orders may be placed 10AM to 5:30PM daily, Mon thru Fri. Call 602-931-4528. Check reader service card or send stamp for our latest flyers packed with new and surplus electronic components.

COMPUTER WAREHOUSE STORE

DEPT. K, 584 COMMONWEALTH AVE.
BOSTON, MA 02215 (617) 261-2701

SYKES COMPUCORDER 100

$950 + $35 SHIPPING

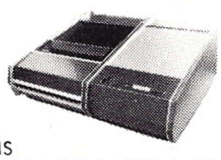
GREAT CASSETTE RECORDER OFFERS 3.6M BIT STORAGE TRANSFER RATE OF 500 CH/SEC AT 1000 BPI. READ WRITE SPEED 5 IPS RECORDING BIT SERIAL. BIPHASE ENCODED WITH VARIABLE BLOCK LENGTHS UNDER PROGRAM CONTROL

TECHTRAN 4100 $595

TAPE CASSETTE DRIVE + $35 SHIPPING U.S.

VERSATILITY PLUS IS YOURS WITH THIS ORIGINAL COST $3200 DRIVE. JUST PLUG IT IN RS232. CAN RUN DIRECTLY FROM TERMINAL INDEPENDENT OF CPU. FULL EDIT CAPABILITY, ALL FUNCTIONS UNDER SOFTWARE CONTROL. LIMITED QUANTITY AVAILABLE

TALLY T132



7 x 8 DOT MATRIX IMPACT PRINTER HAS A SINGLE LINE DYNAMIC MEMORY AND A UNIVERSAL INTERFACE TO ACCEPT PARALLEL DATA, FORMS TO 14-7/8 IN. SIDE, SIMPLE PRINTING MECHANISM USES 132 SOLENOID HAMMERS AND TWO STEPPER MOTORS FOR 100 LPM, 132 COLUMNS, 64 CHARACTERS

$950 + SHIPPING 150 lb.

1-DAY SHIPMENT KITS, MPUs, CPUs (FROM OUR STOCK)

LSI....ADM 3 KIT..........$875
UPPER/LOWER CASE OPTION 100
10 KEY NUMERIC PAD..... 150

IMSAI..8080A KIT 5 SLOT..$599
8080A KIT 22 SLOT.. 651
4K MEMORY KIT...... 139

VIKING 100 PIN CONNECTOR..
HEAVY DUTY.........$ 5

SERIAL I/O KIT..... 125
PROM 4-512 KIT..... 165
UCRI-1 KIT......... 59
CABLE A KIT........ 18

SWTPC..6800 KIT...........$399
MPA.....145 MPB.... 40
MPC..... 40 MPD.... 35
MPE..... 15 MPF.... 30
MPM..... 80 MPMx... 45
MPP..... 35 MPL.... 35
MPS..... 35 MPAb.14.50
MPMb..14.50 MP8b... 30
MPCb,MPSb,MPLb ea. 9.50
CONN. SETS MPU/MEM 2.50
CONN. SETS INTERFC 2
4KBA.... 5 GT61... 99
AC30 AUDIO INTRFC 79.50
PP40.... PRINTER....250
CT 1024 TERM. KIT..275
CTI.....175 KBD 49.95
CTP...15.50 CTS..39.95
CTCA..............15.50
ALL SWTPC UNITS ARE KITS

KIM 1..6502...............$245
KIM 2..4K................. 179
KIM 3..8K................. 289
MANUALS PACKAGE........... 15

ICOM FLOPPIES
FF36-1............$1195
FF36-2............ 1895
360-58............ 300
S171 H............ 250

TARBELL AUDIO
CASSETTE KIT $120

INTERSIL INTERCEPT JR.....$281
12K RAM................. 145
ROM/PROM BOARD 74.65
YOU ADD MEM CHIPS
SERIAL 1/0 ....81.50
AUDIO VISUAL BOARD 125

NATIONAL SEMICONDUCTOR
SC/MP KIT .... $ 99
KEYBOARD KIT 95

SMOKE 16 K MEMORY $595

shipping on kits
under $100....$5
over $100....$10
WRITE FOR OUR COMPLETE CATALOG
covering kits, used equipment and our wide range of available books
$1.00

TO ORDER:

1.. ENCLOSE CHECK FOR FULL PRICE PLUS SHIPPING CHARGES (IN MASS, ADD 5% SALES TAX)
2. CLEARLY IDENTIFY SHIPPING ADDRESS.
3. DESCRIBE ITEM BY MODEL #

1 DAY SHIPMENT IF BANK CHECK OR MONEY ORDER

COMPONENTS FOR SYSTEMS

FEATURES OF THE MONTH

GREEN PHOSPHOR VIDEO MONITOR

$150 + $25 SHIPPING

TOP QUALITY CRTS FROM A MAJOR VENDOR ... NOT JUST A REWORKED TV SET. STANDARD 1V P TO P COMPOSITE VIDEO INPUT, 10 MHz BAND WIDTH, RASTER SCAN 12 x 12 x 13 IN., WITH POWER SUPPLY VIDEO AMPLIFIER, DRIVING CIRCUITRY, VENTILATION MUFFIN FANS, 7 x 9 IN. HORIZONTAL VIEWING AREA UP TO 24 LINES x 80 CHAR., ANTIGLARE ¼ IN. ETCHED GRADIENT DENSITY FACE PLATE, P39 GREEN PHOSPHOR FOR BETTER VIEWING EASE, ON/OFF BRIGHTNESS CONTROLS, 115VAC, 60 W. (SPOT SIZE .015 IN. NOMINAL) ... TRULY A COMMERCIAL UNIT BUILT TO WORK IN A DEMANDING ENVIRONMENT. WE'VE RUN THREE OF THESE OFF OUR SwTPC TERMINAL KIT AT ONCE FOR DEMONSTRATIONS.

ALL an ASR 33 is and MORE!

WE'VE SOLD OUT 3 TIMES ON THIS HEAVY-DUTY TELETYPEWRITER. THIS SHIPMENT IN GREAT CONDITION OFFERS RS232 INTERFACE, QUIET OPERATION, 10 CPS BUILT-IN PAPER TAPE PRINTER/PUNCH, ELECTRIC TYPEWRITER KEYBOARD WITH ADDITIONAL 10 KEY NUMERIC PAD, YOUR CHOICE OF FRICTION OR SPROCKET FEED, LIGHTED PLATEN AREA FOR EASY READING , STANDARD PAPER AND TAPE, SUPPORTED BY OLIVETTI, IMPACT PRINTER GOES UP TO 6 COPIES, VERTICAL SPACING ADJUSTABLE.

$950 + SHIPPING 165 lb

DATAPOINT 3300-200 THERMAL PRINTER

SURPRISING LITTLE THERMAL PRINTER USES WELL RESPECTED AND FIELD PROVEN NCR EMT-1-AE PARALLEL PRINTER WITH ADDITIONAL CIRCUIT BOARDS TO PROVIDE SERIAL RS232 INTERFACE, PRINTS UP TO 30 CPS. 110 VAC PS. USES WIDELY AVAILABLE NCR PAPER, 96 CHAR. ASCII, 80 COL., CRT COMPATIBLE 5 x 7 DOT MATRIX, SOLID STATE WITH LESS THAN 25 MOVING PARTS.

$475 + $25 SHIPPING U.S.

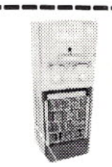
DATAPOINT CASSETTE 3300-300 $195 + $25 SHIPPING U.S.

SMALL STYLIZED CASSETTE RECORDER SERVES AS ADJUNCT BETWEEN CRT TERMINAL AND CPU. ON LINE STORAGE, OFF LINE MESSAGE PREPARATION, 450,000 CHAR. PER CASSETTE, NO POWER SUPPLY, I/O UP TO 2400 BPS.

KLEINSCHMIDT 311 $250 + SHIPPING 75 lb.

THIS 30 CHAR/SEC DRUM PRINTER SITS IN A SOUND-PROOF ENCLOSURE, 64 CHAR., PARALLEL INPUT, 80 CHAR/LINE, ORIGINAL PRICE $2100 WITH ENCLOSURE

DIGITRONICS D507



PAPER TAPE TRANSMITTER
BEAUTIFUL 5' ENCLOSED CABINET

PROVIDES TREMENDOUS SUPPLY OF GOODIES INCLUDING DIGITRONICS 2500 PHOTOELECTRIC PAPER TAPE READER, HEAVY DUTY POWER SUPPLY, 3 MUFFIN FANS, POWER CONTROL PANEL, CIRCUIT BOARDS, RELAYS, CABLES. SOME HAVE PAPER TAPE HANDLERS, ALL ON CASTERS

$95 + SHIPPING 400 lb.

DATAPOINT SERVO PRINTER IN DESK CONSOLE

$395 + SHIPPING 285 lb.

IDEAL UNIT TO BUILD A SYSTEM AROUND. BOTH UNIVAC AND SINGER BUILT THESE PRINTER MECHANISMS WHICH OPERATE AT 30 CPS FROM A ROTATING WHEEL. 65 CHAR. USES STANDARD PRINTOUT OR TYPEWRITER PAPER. PINWHEEL IS INTERCHANGEABLE.

UNIVAC 0769-06 PRINTER MECHANISM ONLY...$295
INCLUDES MOTOR/PRINT WHEEL + SHIPPING 75lb.

VISIT OUR STORE: 9 TO 9 WEEKDAYS; 9 TO 6 SATURDAY

"As the designer of the very first frequency counter, I congratulate you on a very neat design at an excellent price. Performance of this counter is really impressive..."

R. K. Dickey
Professional Electrical Engineer
(Designer of the world's first frequency counter at Berkeley Scientific, 1949)

Thank-you, Mr. Dickey.

Thousands more who own Hufco frequency counters proudly agree.
Since 1974, Hufco has made counters and "counter-offers" that satisfy people to no end. The Hufco combination—a rock-bottom price and sky-high quality—is proving a very popular duo. Here are three good examples why:

A 500 mHz (6-digit) frequency counter for under 40 cents per mHz!

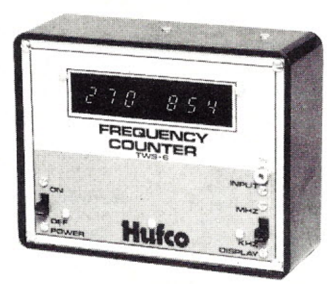
Figure it out. It adds up to unheard-of savings. With guaranteed quality to match and exceed the overpriced brands. Let us prove it to you today.

169.95
(500 mHz kit)
$199.95 assembled

A 30mHz or 250mHz (6-digit) frequency counter — proven best sellers!

69.95
(30mHz kit)
$99.95 assembled

119.95
(250mHz kit)
$139.95 assembled



No wonder people respect the Hufco name.

Incidentally, you'll also find it on economically-priced digital display adapters, voice-operated transmits, power mike adapters, CB/Ham timers, and more.

Interested?
Rush us this coupon today.

Please send me:

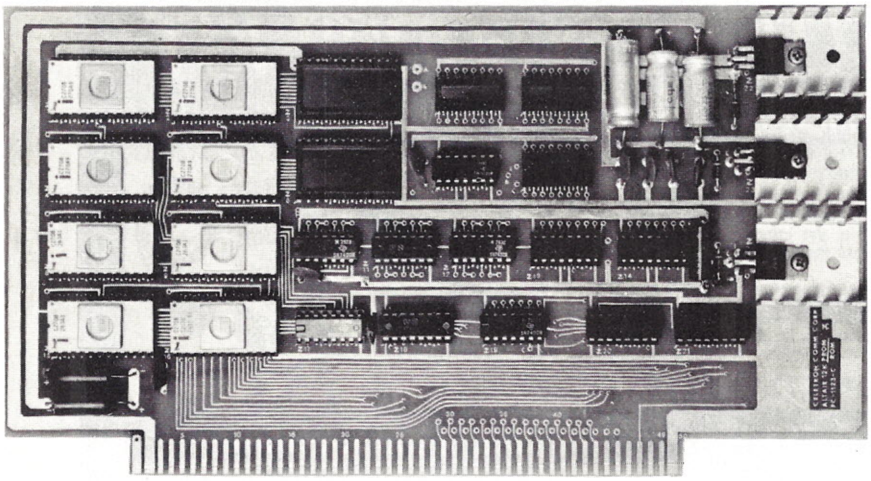
- ☐ **500 mHz frequency counter** - 169.95 kit/199.95 assembled (Enclose check or money order.)
- ☐ **250 mHz frequency counter** - 119.95 kit/139.95 assembled (Enclose check or money order.)
- ☐ **30 mHz frequency counter** - 69.95 kit/99.95 assembled (Enclose check or money order.)
- ☐ **Information on other handy Hufco products.**

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

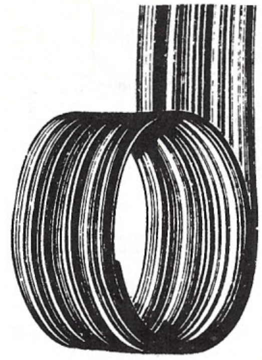
Mail to: Hufco Box 357, Dept. R, Provo, Utah 84601 801/375-8566



SPECTRA FLAT TWIST

50 conductor, 28 gauge, 7 strands/conductor made by Spectra. Two conductors are paired & twisted and the flat ribbon made up of 25 pairs to give total of 50 conductor. May be peeled off in pairs if desired. Made twisted to cut down on "cross talk." Ideal for sandwiching PC boards allowing flexibility and working on both sides of the boards. Cost originally $13.00/ft

SP-324-A $1.00/ft. 10 ft/$9.00



SP-234-A $1.00 ft 50 cond. 10 ft/$9.00
SP-234-B .90 ft 32 cond. 10 ft/$8.00

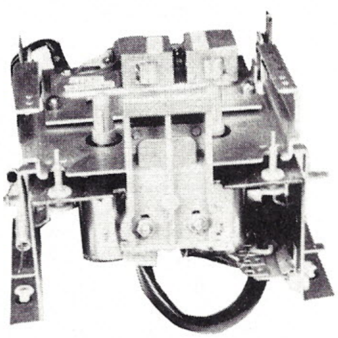
WIRE WRAP WIRE

TEFZEL blue #30 Reg. price $13.28/100 ft. Our price 100 ft $2.00; 500 ft $7.50.

MULTI COLORED SPECTRA WIRE

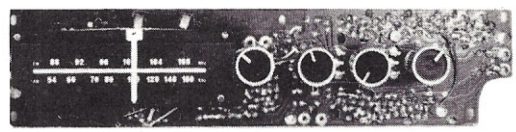
Footage			10'	50'	100'
8	Cond.	#24	$2.50	9.00	15.00
12	"	22	3.00	11.00	18.00
14	"	22	3.50	13.00	21.00
29	"	22	7.50	28.00	45.00

Great savings as these are about 1/4 book prices. All fresh & new.



VIATRON cassette tape deck as shown $35.00
Set of 2 read/write, amplifiers & servo control boards for deck $40.00
VIATRON full spare set of 11 microprocessor boards, checked out . $200.00
COM ADAPTER kit RS-232C will run any baud rate 110-1200 BPS. TTL logic and UART w/diagram, control panel for Viatron & cable $90.00
COM ADAPTER as above fully assembled & tested, panel, cables ... $150.00
VIATRON large tape deck, 7 bit ascii, 9 track, 800 BPI w/power supply cables, control panel, tape, instructions $375.00

AM-FM solid state w/power supply for 115 volts AC. Has mike input, stereo tape or record player input, built-in loop stick. Ready to use, just add speakers $10.00

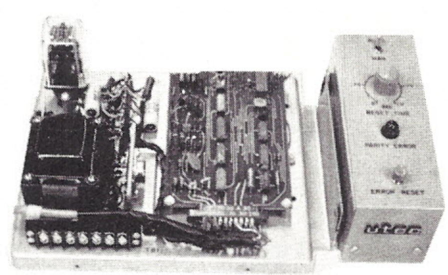


CREDIT VALIDATER

Made for feeding into a computer terminal to query the credit balance of a customer with push-button entry and audio response. Power supply in separate case attached to base may be removed to use for other projects should you wish to scrap. Regular 115 volt input and output of plus 5 volts and plus & minus 12 V both DC & regulated. The phone is touch pad style, automatic electric pad. All sold "as is" 2 PC boards inside jammed with various parts & ICs including memory. Uses buffers ICs to accelerate & decelerate call traffic. Inquiry from Validier emerge as short bursts of Binary data signals. Normally used over leased phone lines at 200 times the capacity of messages. All appear to be excellent & complete. A Computer gadgeteers SPECTACULAR. And at a crazy price of only $18.00. 2 for $35.00
Ship. wgt 10 lb. #SP-153AL ea $18.00

PARITY DETECTOR

New packaged, made for RCA, detects even or odd parity, baud rate 110, 150 or 134.46. Built-in logic supply for the ICs, operates from standard 115 Vac. Control panel allows manual or automatic reset mode of operation. Aluminum enclosure (not shown) covers the electronics. TTY compatible.
Ship wt 10 lbs. ~~$18.50~~
$12.50



Meshna

Please add shipping cost on above.
FREE CATALOG SP-9 NOW READY
P.O. Box 62K, E. Lynn, Massachusetts 01904

NEW CLUBS NOTE!

You could do worse than get on the Kilobaud Klub list. If nothing else this will bring you a free subscription to the Kilobaud Knewsletter, which is worth every cent of the subscription fee. You may also get offers of free equipment from manufacturers . . . but we doubt it. Send the name, address, number of members, when formed, name of secretary, stuff like that to Kilobaud Klub Kaper, Peterborough NH 03458.

GOOD BOOKS?

If you run across a book which you think other hobbyists would profit knowing about, why not write up a brief review . . . including the name and address of the publisher (if you have it) and the price . . . and send it to Kilobaud, Peterborough NH 03458?

The Compucolor 8001 System.

It's A Stand Alone Micro Computer With Color Input/Output Capabilities All In One Package. For Only $2995.

If you're looking for an input device, an output device and a micro computer all in one package, you've found it. The Compucolor 8001. It's here now, in color, on sale for only $2995.

We gave it a memory of its own.

And Floppy Tape Memory is just for starters. Look at these other features. BASIC Language, 8080 CPU, 8 color CRT Terminal, 8K RAM Workspace, Selectable Baud Rate to 9600, Two RS 232 I/O's, Keyboard with 16 Function Keys, Background Color, Lower Case ASCII Characters, Roll, Insert/Delete, 48 Line X 80 Characters/Line, 2X Character Height, thorough operating instructions and a Graphics Mode with 160 X 192 Elements. And our unique Nine Sector Convergence System guarantees you quick set-up, exceptional stability and outstanding color registration in three to five minutes. If you can find a better buy in a color Intelligent CRT and Micro Computer system, let us know. We think we've got the best of both worlds at the best price going. And we want to prove it to you.

Name your game.

After all, you'll have your very own personal computer right at your fingertips. For the most simple or complex tasks. Or just plain fun. The applications are unlimited. Color graphics and computations, check book balancing, educational instruction, tutoring and a unique variety of computer games. Like Star Trek and Hangman and Pong. You can even sit back and enjoy a game of chess. Like we said, the applications are unlimited.

How about a little demonstration?

You'll find a list of our distributors at the bottom of the page. So drop by and ask for a demonstration. Get some answers to your questions. And if you aren't near one of our distributors, give us a call. We've got the answers. The Compucolor 8001. You won't find a better buy in a color CRT Terminal and Micro Computer.

Compucolor Corporation
A subsidiary of
Intelligent Systems Corp. ®

5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

CALIFORNIA
Byte Shop
155 Blossom Hill Rd.
San Jose, Ca. 95123
Larry Grihalva
(408) 226-8383

Computer Store
1093 Mission St.
San Francisco, Ca. 94103
Al Chern
(415) 431-0640

CALIFORNIA
The Computer Center
8205 Ronson Rd.
San Diego, Ca. 92111
Ron Eate
(714) 292-5302

The Computer Mart of Los Angeles
625 W. Katella No. 10
Orange, Ca. 92667
George Tate
(714) 633-1222

GEORGIA
The Computer Systems Center
3330 Piedmont Rd., NE
Atlanta, Ga. 30305
Jim Dunion
(404) 231-1691

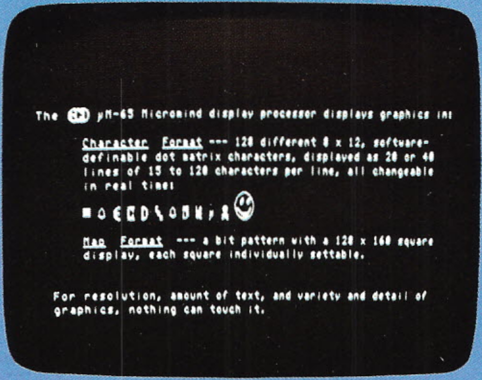
ILLINOIS
Itty Bitty Machine
1316 Chicago Ave.
Evanston, Ill. 60201
Jim Bannish
(312) 328-6800

INDIANA
Home Computer Shop
10447 Chris Dr.
Indianapolis, Ind. 46229
James B. Baughn
(317) 894-3319

MASSACHUSETTS
The Computer Store
120 Cambridge St.
Burlington, Mass. 01803
Sid Halligan
(617) 272-8770

WASHINGTON
Retail Computer Store
410 N.E. 72nd Street
Seattle, Wash. 98115
Tim Broom
(206) 524-4101

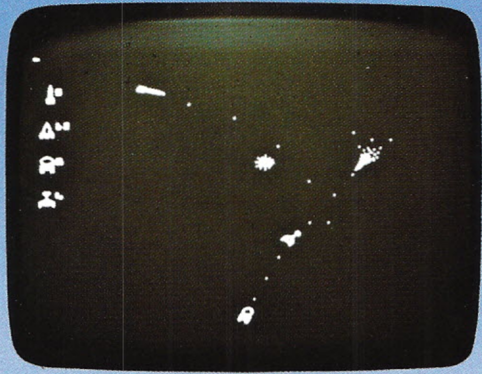
Or Contact Us Direct
5965 Peachtree Corners East
Norcross, Georgia 30071
Telephone (404) 449-5961

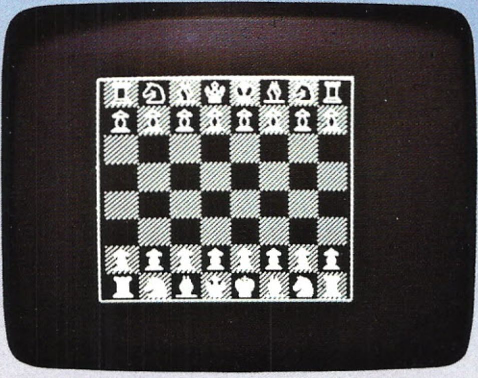


Key Into Maxi-Power @ Micro-Price

Micromind is an incredibly flexible, complete and expandable, hardware/software, general purpose computer system. You won't outgrow it.

Hardware includes an 80 key, software-definable keyboard, I/O interface board, 6500A-series microprocessor (powerful enough for advanced computing), a high-detail graphics and character display processor, power supply, rf modulator, and connections for up to 4 tape recorders plus TV or monitor. An interconnect bus

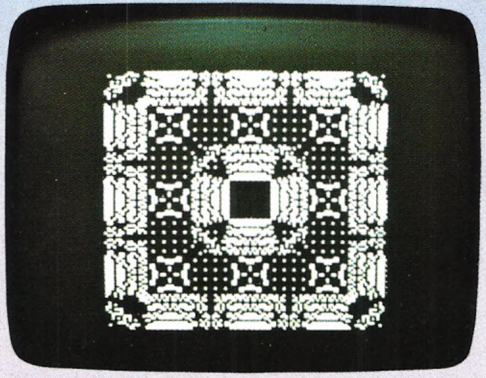




permits 15 additional microprocessors, parallel processing and vastly increased computing power.

System software—including ECD's own notsoBASIC high level language, on advanced error-correcting tape cassettes —provides a word processing editor, a powerful assembler, a debugger, a file system, graphic routines, and peripheral handlers. We also include dynamic graphic games: Animated Spacewar and Life.

ECD's standard Micromind μM-65 supplies 8K bytes of memory. Additional



32K byte expansion boards and a mapping option give Micromind expandable access to 64 Megabytes. Utilizing software-controlled I/O channels, Micromind's advanced encoding techniques load data from ordinary tape recorders at 3200 bits per second.

Micromind comes to you ready-to-use, factory assembled and fully tested. Among microcomputers, it has the largest memory capacity and the fastest storage. You're looking at the work of the finest display processor on the market. You won't find a microcomputer with a more powerful CPU. You won't find a computer with a more flexible keyboard. You won't find anything to touch it at **$987.54.**

So, quit the kluge scene and key into Micromind. You'll be a main frame performer, with all the comforts of home. We're not fooling...this is the cat's μ!

ECD CORP.
196 Broadway, Cambridge, Mass. 02139
(617) 661-4400

ECD

Name ______
Address ______
City/State ______ Zip ______
☐ Fantastic! Check enclosed: **$987.54.** Shipping paid by ECD
☐ BankAmericard ☐ Master Charge Mass. Resident add 5% Sales Tax
______ Expiration Date ______
Signature ______
☐ Send me your brochure.
k

Actual unretouched photographs.

Now you can buy an Altair™ 8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

ALTAIR COMPUTER CENTER
8105 SW Nimbus Ave.
BEAVERTON, OR 97005

COMPUTER KITS (S.F. area)
1044 University Ave.
BERKELEY, CA 94710
(415)-845-5300

THE COMPUTER STORE
(Arrowhead Computer Co.)
820 Broadway
SANTA MONICA, CA 90401
(213)-451-0713

GATEWAY ELECTRONICS, INC.
OF COLORADO
2839 W. 44th Ave.
DENVER, CO 80211
(303)-458-5444

COMPUTER SHACK
3120 San Mateo N.E.
ALBUQUERQUE, NM 87110
(505)-883-8282; 883-8283

ALTAIR COMPUTER CENTER
4941 East 29th St.
TUCSON, AZ 85711
(602)-748-7363

ALTAIR COMPUTER CENTER
611 N. 27th St. Suite 9
LINCOLN, NB 68503
(402) 474-2800

COMPUTER PRODUCTS UNLIMITED
2412 Broadway
LITTLE ROCK, AR 72206
(501)-371-0449

ALTAIR COMPUTER CENTER
110 The Annex
5345 East Forty First St.
TULSA, OK 74135
(918)-664-4564

ALTAIR COMPUTER CENTER
5750 Bintliff Drive
HOUSTON, TX 77036
(713)-780-8981

COMPUTERS-TO-GO
4503 West Broad St.
RICHMOND, VA 23230
(804)-335-5773

MICROSYSTEMS (Washington, D.C.)
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703)-569-1110

THE COMPUTER STORE
Suite 5
Municipal Parking Building
CHARLESTON, W. VA. 25301
(304)-345-1360

THE COMPUTER ROOM
3938 Beau D'Rue Drive
EAGAN, MN 55122
(612)-452-2567

THE COMPUTER STORE
OF ANN ARBOR
310 East Washington Street
ANN ARBOR, MI 48104
(313)-995-7616

THE COMPUTER STORE, INC.
(Hartford area)
63 South Main Street
WINDSOR LOCKS, CT 06096
(203)-627-0188

CHICAGO COMPUTER STORE
517 Talcott Rd.
PARK RIDGE, IL 60068
(312)-823-2388

GATEWAY ELECTRONICS, INC.
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314)-427-6116

BYTE'TRONICS
Suite 103
1600 Hayes St.
NASHVILLE, TN 37203
(615)-329-1979

THE COMPUTER STORE, INC.
120 Cambridge St.
BURLINGTON, MA 01803
(617)-272-8770

ALTAIR COMPUTER CENTER
269 Osborne Road
ALBANY, NY 12211
(518)-459-6140

THE COMPUTER STORE
OF NEW YORK
55 West 39th St.
NEW YORK, NY 10018
(212)-221-1404

THE COMPUTER SYSTEMCENTER
3330 Piedmont Road
ATLANTA, GA 30305
(404)-231-1691

MARSH DATA SYSTEMS
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813)-886-9890

mits